Efficient large-context dependency parsing and correction with distributional lexical resources

Enrique Henestroza Anguiano

Université Paris Diderot (Paris 7)
École Doctorale de Sciences Mathématiques de Paris-Centre n°386

Doctorat d'Informatique

ENRIQUE HENESTROZA ANGUIANO

Efficient Large-Context Dependency Parsing and Correction with Distributional Lexical Resources

Analyse syntaxique probabiliste en dépendances : approches efficaces à large contexte avec ressources lexicales distributionnelles

Thèse sous la direction de :

Laurence DANLOS, Alexis NASR, et Marie CANDITO

Soutenue le 27 juin 2013

Composition du jury :

M. Bernd BOHNET, examinateur
Mme Marie CANDITO, co-directrice de thèse
M. Matthieu CONSTANT, rapporteur
Mme Laurence DANLOS, directrice de thèse
M. Alexis NASR, co-directeur de thèse
M. Joakim NIVRE, rapporteur

Abstract

This thesis explores ways to improve the accuracy and coverage of efficient statistical dependency parsing. We employ transition-based parsing with models learned using Support Vector Machines (Cortes and Vapnik, 1995), and our experiments are carried out on French.

Transition-based parsing is very fast due to the computational efficiency of its underlying algorithms, which are based on a local optimization of attachment decisions. Our first research thread is thus to increase the syntactic context used. From the arc-eager transition system (Nivre, 2008) we propose a variant that simultaneously considers multiple candidate governors for right-directed attachments. We also test parse correction, inspired by Hall and Novák (2005), which revises each attachment in a parse by considering multiple alternative governors in the local syntactic neighborhood. We find that multiple-candidate approaches slightly improve parsing accuracy overall as well as for prepositional phrase attachment and coordination, two linguistic phenomena that exhibit high syntactic ambiguity.

Our second research thread explores semi-supervised approaches for improving parsing accuracy and coverage. We test self-training within the journalistic domain as well as for adaptation to the medical domain, using a two-stage parsing approach based on that of McClosky et al. (2006). We then turn to lexical modeling over a large corpus: we model generalized lexical classes to reduce data sparseness, and prepositional phrase attachment preference to improve disambiguation. We find that semi-supervised approaches can sometimes improve parsing accuracy and coverage, without increasing time complexity.

Résumé

Cette thèse présente des méthodes pour améliorer l'analyse syntaxique probabiliste en dépendances. Nous employons l'analyse à base de transitions avec une modélisation effectuée par des machines à vecteurs supports (Cortes and Vapnik, 1995), et nos expériences sont réalisées sur le français.

L'analyse a base de transitions est rapide, de par la faible complexité des algorithmes sous-jacents, eux mêmes fondés sur une optimisation locale des décisions d'attachement. Ainsi notre premier fil directeur est d'élargir le contexte syntaxique utilisé. Partant du système de transitions arc-eager (Nivre, 2008), nous proposons une variante qui considère simultanément plusieurs gouverneurs candidats pour les attachements à droite. Nous testons aussi la correction des analyses, inspirée par Hall and Novák (2005), qui révise chaque attachement en choisissant parmi plusieurs gouverneurs alternatifs dans le voisinage syntaxique. Nos approches améliorent légèrement la précision globale ainsi que celles de l'attachement des groupes prépositionnels et de la coordination.

Notre deuxième fil explore des approches semi-supervisées. Nous testons l'autoentrainement avec un analyseur en deux étapes, basé sur McClosky et al. (2006), pour le domaine journalistique ainsi que pour l'adaptation au domaine médical. Nous passons ensuite à la modélisation lexicale à base de corpus, avec des classes lexicales généralisées pour réduire la dispersion des données, et des préférences lexicales de l'attachement des groupes prépositionnels pour aider à la désambiguïsation. Nos approches améliorent, dans certains cas, la précision et la couverture de l'analyseur, sans augmenter sa complexité théorique.

Acknowledgements

I would like to warmly thank my three advisors, who helped me at every stage to make the completion of my thesis possible. Laurence Danlos graciously welcomed me into the fold of the Alpage research group at the Université Paris Diderot and the INRIA, and provided much needed motivation to complete my thesis in a timely manner. Alexis Nasr took a chance in bringing me on as a member of the SEQUOIA project, and was an invaluable source of guidance in plotting the course of my research. Marie Candito, in particular, has been a great mentor and teacher, whose insights and assistance on matters academic and beyond allowed me to thrive in a new country and setting; I thank her for her insistence on intellectual rigor, her endless patience in explaining aspects of French linguistics, and her willingness to meet over coffee even during her busiest moments. This thesis would also not have been possible without the funding of my doctorate by the Agence Nationale de la Recherche, as part of the SEQUOIA project ANR-08-EMER-013.

Joakim Nivre, Matthieu Constant, and Bernd Bohnet graciously agreed to carefully read my thesis and serve as members of the jury. I thank them for their insights and constructive feedback.

I was lucky to be a part of the Alpage research group, which provided an intellectually stimulating and collaborative environment. I got to know and appreciate every member; in particular, Djamé Seddah was a helpful colleague who provided useful insights on how to tackle treebank and tagging issues, while Pascal Denis was a source of great discussions on machine learning as well as slightly less enjoyable thrashings on the tennis court. And I couldn't have survived without my fellow doctorate students, who I thank for welcoming me with open arms and an eternal willingness to explain French jokes to a clueless American.

Everyone who I had the chance to meet and befriend during my three years in Paris, I thank you for showing how this city can be made even more beautiful by the people with whom you surround yourself. Expats, thank you for opening my eyes to the many different cultures and walks of life from which we all come, as well as for the much needed reminders of home. La troupe, je ne crois pas à la chance que j'ai eu de vous rencontrer; il est difficile d'exprimer à quel point je vous apprécie.

Finalmente, a mi mamá, mi papá, y mi hermanita Moni, les quiero decir que todo lo que he logrado ha sido gracias a su apoyo.

Contents

List of Figures

List of Tables

Introduction

Positioning our Research

The primary objective of this thesis is to study ways to improve wide-coverage syntactic parsing, with French as our language of application. Given the openness of this objective, as well as the extensive amount of research on parsing that has been conducted over the decades within the field of Natural Language Processing (NLP) for many languages, our process involved narrowing the scope of our investigation to a specific framework and then seeking to make focused contributions aimed at improving methods within that framework.

A major decision we make in our choice of framework is to favor statistical parsing approaches over symbolic ones. One major advantage of statistical parsing is its robustness, with an ability to predict the syntactic structure of sentences that may not adhere strictly to the grammar of a language. Another advantage is its inherent ability to disambiguate between alternative syntactic parses for a given sentence. The downsides stem primarily from the fact that the most accurate statistical parsers are trained in a supervised manner over manually-annotated treebanks. One consequence is the *data sparseness* problem, where a statistical parser captures lexical information only for frequent phenomena in the limited-size treebank. Another consequence is that the parser is tied to the specific genre or domain of the treebank, which is usual composed of journalistic text; the parser may have difficulty generalizing to other domains, such as biomedical text or user-generated web content. A major concern of our thesis is to address these problems, using semi-supervised methods to mitigate data sparseness and improve coverage beyond the journalistic domain.

The other major decision we make in our choice of framework is to favor syntactic dependency grammar and representation over the more typically used phrase-

structure grammar and representation. We are primarily guided by the existence of linear-time dependency parsing algorithms with competitive accuracy compared to the best state-of-the-art phrase-structure parsing algorithms, which have a complexity of cubic time or more. Another advantage is that direct syntactic dependencies between words are arguably closer to the predicate-argument structure required for semantic processing downstream.

These decisions ultimately lead us to a computationally efficient framework that includes data-driven transition-based dependency parsing, which has developed rapidly over the past decade (Nivre, 2003; Yamada and Matsumoto, 2003; Nivre et al., 2006; Nivre, 2008; Kübler et al., 2009), and the less studied dependency parse correction (Hall and Novák, 2005; Attardi and Ciaramita, 2007). While we will describe this framework extensively later on, we note for now that it allows for the development of parsers and correctors that have the qualities of compactness and computational efficiency we believe are necessary for real world applications.

Working within this framework, our research is divided into two main threads, whose shared goal is to improve parsing accuracy and extend coverage while retaining computational efficiency. The first research thread investigates methods that introduce additional syntactic context into attachment decisions for both parsing and correction, culminating with their combination into a two-stage parser that is trained in a semi-supervised manner for both in-domain and out-of-domain parsing. The second research thread continues on the semi-supervised path, investigating the integration into parsing model features of distributional lexical resources built from corpora that have been automatically-parsed with the two-stage parser, and again considering the use of these methods for both in-domain and out-of-domain parsing.

Outline of the Thesis

The body of this thesis is divided into five chapters. A preliminary chapter first presents the essential linguistic and mathematical notions required for later investigations into dependency parsing and working with lexical resources. Subsequent chapters elaborate the two main threads of our research, with each thread associated with two chapters and an overall progression in which the algorithms and results from previous chapters inform the course of later ones.

Chapter 1

The chapter "Preliminaries in Syntax and Machine Learning" is dedicated to laying the theoretical groundwork for our thesis as well as presenting the primary materials — the core data set and computational modeling techniques — that we use in

our experiments. Our first objective in this chapter is to provide a theoretical linguistic background to our work, in particular deciding what constitutes a valid and useful syntactic representation of a sentence. Given the variable nature of linguistics terminology across different theories, we seek to provide operational definitions for the linguistic concepts we need in this thesis. We also pay special attention to French syntax, describing the syntactically annotated resource that we use to train and evaluate our parsing models. Our second objective in this chapter is to detail the fundamental machine learning algorithms that are at the core of our automatic syntactic parsing approaches.

In Section 1.1 we begin by introducing the basic linguistic notions of distribution, morphology, word forms, and lexemes, which are important for understanding syntactic theory, and then introduce the notion of syntax. In Section 1.2 we move into the heart of our discussion of syntax by presenting the two major formalisms, phrase-structure grammar and dependency grammar. We explain the equivalence, under certain conditions, of these two formalisms, and justify our decision to work exclusively with dependency grammar. In Section 1.3 we move from a general discussion of syntax to the specific case of French. We present the syntactically annotated resource at our disposal, the French Treebank, with a discussion of important linguistic decisions present in its annotation as well as its conversion from phrase-structure to dependencies. Finally, in Section 1.4 we delve into the fundamental computational modeling aspects of the thesis. We briefly introduce machine learning, and then describe the supervised learning algorithms we use to train linear models of classification and ranking, which drive the syntactic parsers and correctors used in later chapters.

Chapter 2

Having covered fundamental aspects of syntactic dependency tree representation and machine learning algorithms, the chapter "Efficient Large-Context Dependency Parsing" formally describes the problem of dependency parsing and begins investigating the first main thread of our thesis, which involves the search for methods that introduce additional syntactic context into attachment decisions. We present empirical results for various classes of parsers, settle on the efficient transition-based family of parsers, and introduce a variant transition system that introduces more context into dependency attachment decisions.

In Section 2.1 we formalize the problem of dependency parsing and investigate practical tradeoffs between algorithmic efficiency and the amount of contextual information available for dependency attachment decisions. To help guide this investigation, we describe a preliminary study in which different dependency parsing

approaches were tested for French, providing empirical evidence supporting our decision to focus exclusively on computationally efficient transition-based parsing. In Section 2.2 we present the family of transition-based parsing algorithms that have been popularized in the past decade and discuss the existing ARC-STANDARD and ARC-EAGER algorithms within this family. We then introduce ARC-EAGER-MC, a large-context variant that simultaneously considers multiple candidate governors for certain attachments during parsing. Finally, Section 2.3 describes parsing experiments for French in which we compare the performance of ARC-STANDARD, ARC-EAGER, and ARC-EAGER-MC, with a particular focus on the parsing accuracy for PP-attachment and coordination, two ambiguous syntactic constructions that are typically difficult to parse.

Chapter 3

The chapter "Efficient Large-Context Parse Correction" continues the first main thread of our thesis, building on the methods and results from the previous chapter by investigating the use of second-stage parse correction on top of first-stage transition-based parsing. In the previous chapter, we incorporate directly into the parser's transition system a way to consider multiple candidate governors simultaneously for a given dependent. In this chapter, we stay with a traditional transition system and instead seek to incorporate additional syntactic context in a second-stage correction model that revises attachments; parse correction has the dual benefits of more surrounding syntactic context and the ability to simultaneously compare multiple candidate governors for a dependent in a computationally efficient manner. We also test a semi-supervised approach to two-stage parsing aimed at reducing data sparseness and improving coverage for the medical domain.

In Section 3.1 we present the problem of learning to correct dependency parse errors, describing an existing neighborhood correction framework as well as our innovations in using a ranking learner and accessing more syntactic context. In Section 3.2 we discuss an interesting self-training approach to learning with partially unannotated data in a two-stage parsing system, which has previously been used successfully with a phrase-structure parser coupled with a reranker. We investigate a corresponding approach for dependency parsing that couples a transition-based parser with a corrector, and also note how this can be applied to the problem of domain adaptation. Finally, Section 3.3 includes correction experiments for French in which we compare the performance of baseline ARC-STANDARD and ARC-EAGER parsers from Chapter 2 to two-stage systems that additionally correct dependency errors from the parsing stage, with special attention again given to the difficult attachment types of PP-attachment and coordination. We also evaluate to what

extent the self-training approach is effective for two-stage dependency parsing, for both in-domain journalistic text and out-of-domain medical text.

Chapter 4

Having finished our investigation into the first main thread of our thesis and established an efficient base two-stage parsing system, in the chapter "Parsing with Generalized Lexical Classes" we turn to the second main thread of our thesis, which involves the integration of automatically built distributional lexical resources to improve dependency parsing. Using our two-stage parser as a backbone, in this chapter we specifically investigate the semi-supervised creation of generalized lexical classes that can replace word forms in features during parsing. In doing so, our goal is to tackle the problem of accurately modeling lexical relationships from potentially sparse counts in a training corpus.

In Section 4.1, we describe the methods in distributional lexical semantics that create from a large text corpus a distributional thesaurus, which is needed for the creation of our generalized lexical classes. We provide an overview of standard methods used in the NLP literature, leading to a description of different metrics for weighting contexts and calculating similarity that we use to build distributional thesauri. We then describe three spaces of generalized lexical classes we have chosen to investigate, each relying on a distributional thesaurus: (i) lemmas, with word forms mapped to one or more lemmas ranked according to distributional similarity; (ii) clusters, generated using hierarchical agglomeration of word forms based on their pairwise distributional similarities; and (iii) synsets, obtained using automatic sense ranking to score the semantic senses of word forms according to a semantic resource and a distributional thesaurus. Finally, in Section 4.2 we present experiments for French where we test the inclusion in parsing and correction models of lexical features derived from the three spaces of generalized lexical classes.

Chapter 5

In the final chapter, "Parsing with PP-Attachment Preferences," we finish the second main thread of our thesis, borrowing from the distributional methods from the previous chapter and this time investigating semi-supervised models of lexical preference for PP-attachment decisions in parse correction. We focus on PP-attachment because it is known to be difficult to disambiguate syntactically, and we find earlier that it contributes to a large proportion of parsing errors.

In Section 5.1, we start by motivating our investigation of lexical preference for the problem of PP-attachment and discussing the ways in which lexical preference

has been used in the literature for parsing and other NLP applications. We then present an overview of the types of lexical preference we consider for PP-attachment, distinguishing between different levels of lexical specificity in the dependent PP: we consider a less lexically-specified level in line with the notion of subcategorization, and a more lexically-specified level that additionally includes the lemma of the PP's object. We then turn to a discussion of the two statistical metrics we use to obtain preference scores from a large automatically-parsed corpus: (i) a traditional pointwise mutual information metric; and (ii) a novel neighborhood-based relative frequency metric that uses information concerning the syntactic neighborhood of candidate governors surrounding a PP dependent within a parse tree. In Section 5.2 we then present parse correction experiments for French in which we test the addition of preference features for PP-attachment. The evaluation compares the two types of lexical preference and the two statistical metrics used to obtain preference scores, with an additional setting that derives 0-1 scores from a hand-built verb subcategorization resource.

Chapter 1

Preliminaries in Syntax and Machine Learning

"No, no!" said the Queen. "Sentence first — verdict afterwards."

— Lewis Carroll

In this thesis, we investigate approaches to improving upon existing machine learning methods for syntactic natural language parsing. In order to motivate our approaches, it is necessary to present relevant background to our work, as well as the primary materials — data sets and computational modeling techniques — that we use in our research. Our first objective in this chapter is to provide a theoretical linguistic background to our work, and in particular decide on what constitutes a valid and useful syntactic analysis of a sentence. In addition, we need to take into account the particularities of French, the language we have chosen to work with, and the specific treebank resource available to us for our experiments. A second objective in this chapter is to subsequently detail the fundamental machine learning algorithms and methods that are at the core of our statistical approaches to syntactic parsing.

In Section 1.1 we begin our discussion by introducing the basic linguistic notions of distribution, morphology, word forms, and lexemes, which are important for understanding syntactic theory, and then introduce the notion of syntax.

In Section 1.2 we move into the heart of our discussion of syntax, presenting the two major formalisms for representing natural language syntax, which are phrase-structure grammar and dependency grammar. We explain the equivalence, under certain conditions, of the two formalisms, and justify our decision to work exclusively with dependency grammar.

In Section 1.3 we move from a general discussion of syntax to the specific case of French. We present the syntactically annotated resource at our disposal, the French Treebank, with a discussion of important linguistic decisions present in its annotation as well as its conversion from phrase-structure to a version with dependencies that we use for the parsing experiments in this thesis.

In Section 1.4 we finally delve into the fundamental computational modeling aspects of the thesis. We introduce machine learning, and then describe the supervised learning algorithms we use to obtain linear models of classification and ranking, which drive the syntactic parsing and correction models that are used in later chapters.

1.1 Basic Linguistic Concepts

We begin by presenting the elementary linguistic notions underlying natural language syntax, as well as some of the essential characteristics of syntactic relationships between words in a sentence. Starting with these building blocks, we set the stage for a more detailed discussion of grammatical formalisms in the subsequent section. We take care here to use definitions and terminology that are widely agreed

upon in the field of linguistics, and we avoid delving into unnecessarily fine detail while nonetheless ensuring that our definitions are specific and operational.

1.1.1 Morphological and Lexical Notions

We begin by working our way from the smallest linguistic unit of interest to us, the morpheme, up to the word form, which is the key unit in syntax. At these lower levels, meaning necessarily plays an important role in our definitions of units. However, we reiterate that syntax is concerned with communicative form, and therefore our recourse to meaning is limited to this section.

Morphemes and Word Forms

The first approximation of structure in written or spoken communication is considered to be the *utterance*, which consists of a fixed sequence of units of minimal sense. This first minimal meaningful unit of natural language is termed the *morpheme*; more precisely, it is a meaningful unit that cannot be further broken down into smaller meaningful units. For an example in French, 'pomme' ('apple') is a morpheme while 'pommes' ('apples') is not, because the latter can be decomposed into the morpheme 'pomme' and the pluralizing morpheme '–s'. Though 'pomme' can be broken down into letters, syllables, phonemes, or possibly other smaller units, crucially these smaller units are not viewed as carrying meaning.

It is important at this point to begin examining the constraints that exist concerning morpheme combination; specifically, we note that all morphemes cannot be combined with the same amount of freedom. For example, the French prefix morpheme 'in-' (the same as in English) and the morpheme 'petit' ('small') have fairly well-defined meanings when used in combination with other morphemes, with 'in-' indicating opposition and 'petit' indicating small size. However, of these two only 'petit' is considered to be *free*, as it can precede almost any morpheme with a compatible meaning, such as 'chien' ('dog') or 'échec' ('failure'). The morpheme 'in-', on the other hand, is considered to be *bound*, as it can combine with a small number of morphemes with compatible meaning, such as 'in-fini' ('infinite') or 'in-juste' ('unjust'), while not with many others, such as the plausible but not part of the language 'in-grand' ("not large").

The observation that morphemes can have constraints regarding which other morphemes they combine with is encompassed by the larger notion of *distribution*, which operates at various linguistic levels and groups units into classes based on shared sets of observed constraints. Distribution as a linguistic concept is widely used and can be traced back to the seminal work of Bloomfield (1933), who studied

the distribution of morphemes and larger linguistic units. While the morphological level cannot be described with distribution alone, as meaning plays a fundamental role, our later description of syntax will rely primarily on distribution.

The fact that there are differences in the distributional constraints for different morphemes leads us to the next higher level linguistic unit. This unit is the *word form*, which we define as a unit composed of one or more morphemes that is both free and minimal, the latter indicating that it cannot be decomposed into free parts while retaining its meaning. As a consequence of requiring that word forms be free, bound morphemes are no longer stand-alone units at this level, as is evident in the two distinct word forms 'juste' ('just', 'fair') and 'injuste' ('unjust'). The requirement that words be minimal is necessary in order to set a boundary on this level, since otherwise any arbitrarily long sequence of morphemes could be considered a word form. While it is true that in orthographic representations of languages like English and French blank spaces and punctuation marks are good indicators of word form boundaries, this is not the case for the *compound word*, or word form consisting of multiple free morphemes whose meaning is nonetheless non-compositional. While compound words consist of multiple parts that could count as word forms in other contexts, those parts lose their meaning within the compound or get a more specific meaning. For example, the compound word "carte bleue" ('card' and 'blue'), when referring to a credit card, has a unique meaning that is lost if you compose the meanings of 'carte' and 'bleue' separately. The dual criteria for word forms of being free and minimal are thus more precise than orthographic criteria, and in any case more useful when taking into account both spoken and written communication.

Classes of Morphemes

Having made the distinction between morphemes and stand-alone word forms, we can now look back at morphemes and note that they can be placed into two major classes: grammatical and lexical.

Grammatical morphemes constitute a closed class containing a small and stable set of morphemes, whose meanings are either very general or internal to the language itself. A first major type of grammatical morpheme that has a very general meaning is the *inflectional* morpheme, which is always bound and which carries categorical information organized into paradigms, or sets of mutually exclusive values. For example, in the linguistic paradigm of gender, a word form can carry either a masculine or a feminine inflection, but not both. Inflectional morphemes are additionally characterized by the fact that they modify neither the underlying semantic nature of the word form to which they are affixed nor the basic distributional properties of that word form.

The next major type of grammatical morpheme is the *derivational* morpheme, which includes all remaining bound morphemes that are not inflectional. Derivational morphemes are not organized into paradigms and they do modify the underlying meaning of the word to which they are affixed, changing the class of referents denoted. They often also change its basic distributional properties, both in terms of how that word form can be modified by additional morphemes and how that word form can be combined at higher syntactic levels with other word forms. Since derivational morphemes operate at the intersection of morphology and syntax, we will return to them a bit further on in our discussion.

A last key type of grammatical morpheme encompasses those morphemes that operate as connectors. These morphemes are free, appearing as standalone word forms, and have meanings that are primarily internal to the language. An example of this type of grammatical morpheme is the complementizer 'que' ('that'), with additional ones including pronouns, articles, and prepositions, among others.

The second major class of morphemes are called *lexical morphemes*, and they are characterized in opposition to grammatical morphemes through their richer meanings as well as the fact that they constitute an open class. Lexical morphemes are the part of a language that exists in state of flux, with morphemes entering and leaving the language as new concepts in the world are encountered or obsolete ones are discarded. They often have meanings that encompass concepts such as entities, objects, activities, processes and descriptions. An example is the aforementioned morpheme 'pomme'. Though lexical morphemes are largely free, there are some examples of bound lexical morphemes, such as the French prefix 'hipp-', which indicates the quality of a horse, that is found for instance in the word forms 'hippique' ('horse-like') and 'hippocampe' ('hippocampus').

The Lexicon

The word forms of a language are the main units we are interested in, and we introduce a few more terms regarding their use. We define a *lexeme* to be a set of word forms that differ only through inflection. One example would be the set of all inflections of the verb 'manger' ("to eat"), with different values for the grammatical categories that are applicable to verbs: {'mange', 'mangeons', 'mangerait', etc.}. In order to simplify the representation of such sets, a *lemma* for each lexeme is selected as its citation form, and this is usually the form that is minimally inflected. For example, supposing we choose to use infinitive verb forms as lemmas, then 'manger' would be the lemma for its corresponding lexeme. Finally, the *lexicon* for a language is a resource describing each lexeme in the language, including its lemma and the inflectional possibilities leading to its different observed word forms.

1.1.2 Syntactic Notions

The next natural step in our discussion is to look at the interactions and relationships that arise between separate word forms that appear sequentially. *Syntax* is the term used to designate the study of the organizational system governing the relationships between word forms in a language. The syntactic analogue to the utterance, which we described before as a fixed sequence of units of minimal sense, is the *sentence*, a more widely-used term that generally refers to a fixed sequence of word forms. It is complicated, however, to precisely delimit the boundaries of an utterance or a sentence. While this subject is debated in linguistic theory, we find it to be beyond the scope of our work to discuss it here. Instead, we sidestep the issue by noting that since we work on written text, we are able to rely on orthographic markers of sentence boundaries (e.g. periods, exclamation marks, question marks). Supposing that we have such a means of delimiting sentence boundaries, the study of syntax is then traditionally limited to relationships between word forms within a sentence, with relationships that cross sentence boundaries studied at a higher *discourse* level of linguistic analysis.

We have already hinted at the compositional semantics of word forms when noting that compound words such as "carte bleue" are indivisible for reasons pertaining purely to meaning. However, it is important to note that syntax has the quality of being powerful and descriptive enough that its rules governing the relationships between word forms can be explored and understood without recourse to semantics. Our discussion from this point forward will thus use distributional, rather than semantic, properties of language to describe the various components of syntax.

Part-of-Speech Categories

As we transition from a discussion of morphology to one of syntax, a crucial next step is the classification of word forms, or more specifically lexemes, into *part-of-speech (POS) categories*, which identify shared morpho-syntactic behavior. Linguistic theory across languages identifies POS categories as an important bridge between morphology and syntax: lexemes can be grouped by shared behavior in terms of both inflection (morphology) as well as word-to-word relationships (syntax). The major classic POS categories can be divided into open categories, which are comprised of lexemes that have at least one lexical morpheme, and closed categories, whose lexemes are made up of grammatical morphemes only. For both English and French the open categories are verbs, nouns, adjectives, and adverbs, while the closed categories include determiners, prepositions, and conjunctions, among others.

We noted earlier that derivational morphemes may alter a word form's distributional properties, and now we can make this description more precise: a derivational morpheme may change the POS category of the word form it is affixed to. For example, the French derivational morpheme '-able' can transform a verb into an adjective, as can be seen with the verb 'jouer' ("to play") and the noun 'jouable' ('playable'). This change in POS category results in new inflectional constraints, as well as new distributional syntactic properties. It should be noted that derivational morphemes do not always alter a word form's distributional properties, as is the case for the French diminutive morpheme '-ette': both 'maison' ('house') and 'maisonnette' ("small house") are common nouns.

Since we view POS categories as being distributional in nature, we can define them at different levels of granularity: A *coarse-grained* categorization is one with fewer categories and looser distributional alignment between lexemes in a given category, while a *fine-grained* categorization has more categories and tighter distributional alignment between lexemes in a given category. For example, in French the grouping of all nouns into a single category may be viewed as coarse-grained, and a finer-grained categorization would distinguish between common nouns like 'pomme' and proper nouns like 'Jean'. This is because the distributions of common and proper nouns differ slightly, for example with respect to determiners: "la pomme" ("the apple") occurs in French, but excepting some dialect variants no determiner can be associated with 'Jean'.

At this point, we have now described all of the essential linguistic units pertinent to syntax that we will use and refer to throughout this thesis. To summarize and reinforce our understanding of the organization and composition of these linguistic units, we show in Table 1.1 a linguistic analysis at the sentential, lexical, and morphological levels for the French sentence "Nous observions attentivement un arc-en-ciel." ("We were attentively observing a rainbow.").

Grammars and Acceptability

We now look at the broader syntactic notions of grammar and grammaticality. A natural language allows for the expression of an infinite variety of sentences from a finite lexicon, although at the same time there exist many possible sequences of word forms that are not considered part of the language. A *grammar*, in the broad linguistic sense, can then be seen as the set of rules and constraints that dictates which sentences are syntactically valid for a particular language; those sentences that adhere to these rules are considered grammatical, while those that do not are considered ungrammatical.

We stated earlier that our view of syntax is essentially decoupled from mean-

Sentence:	Nous	observions	attentivement	un	arc-en-ciel	.
Lexical:						
Word Form	[nous]	[observions]	[attentivement]	[un]	[arc-en-ciel]	
Part-of-Speech	[pronoun]	[verb]	[adverb]	[determiner]	[noun]	
Morphological:						
Free	[il]	[observer]	[attentif]	[un]	[arc][en][ciel]	
Derivation			[–ment]			
Inflection	[plural] [1st person]	[plural] [1st person] [indicative] [imperfect]		[singular] [male]		

Table 1.1: Sentential, lexical, and morphological break down for the sentence: "Nous observions attentivement un arc-en-ciel." ("We were attentively observing a rainbow.")

ing, due to the fact that the syntax for a language can be studied and validated independently. The reason for this lies in the distinction between grammaticality, which is the adherence of a sentence to the syntax of a language, and *acceptability*, which additionally requires that the sentence be interpretable to a speaker of the language. Using the famous example of Chomsky (1957), a semantically uninterpretable sentence in English such as "Colorless green ideas sleep furiously" is nonetheless considered intuitively grammatical by native speakers of English. Semantic requirements can thus be characterized as rules that distribution-based generalizations cannot capture; from a distributional perspective, the adjective and noun POS categories are observed as appearing together in general, but from a semantic perspective the specific adjective 'green' and noun 'ideas' are not compatible. One can also see the distinction between syntactic and semantic requirements in the case of verb arguments. Transitive verbs such as the French 'attraper' ("to catch") are said to *subcategorize* for, or syntactically require, a nominal object, because such a generalization can be made about this particular class of verbs based on their distributions. Semantically, on the other hand, 'attraper' additionally requires that the object be such that the two combine to create an interpretable meaning, as in "attraper le ballon" ("to catch the ball").

For the purposes of our thesis, it is convenient to maintain a clear divide between grammaticality and ungrammaticality. However, it should be noted that recent trends in linguistics tend toward a more nuanced approach to grammar, arising from easier access to sophisticated search techniques over attested examples in large corpora. The notion of preference has been studied as an alternative to binary grammaticality, for instance in the work of Bresnan et al. (2007) for predicting dative alternation. Looking ahead to the problem of syntactic parsing, we note that our goal is to produce explicit syntactic structures for observed sentences, whether

they be perfectly grammatical or not, and the statistical approaches to parsing we investigate are particularly suited to a non-binary conception of grammaticality.

Grammatical Function and Valency

Before we turn to a discussion of specific grammar formalisms, we introduce two major aspects of syntactic relations that are linguistically important, independently of the formalism used. The first is the notion of *grammatical function*, in which a relationship between words is characterized using a set of grammatical properties such as position, acceptable POS categories, and inflection. An example of an important grammatical function is that of *subject* in French: its properties include the possibility of appearing before the verb, the requirement of inflectional agreement with the verb, the possibility of undergoing relative pronominalization with the pronoun 'qui' ('who'), etc. Specific instances of grammatical relations can then be classified into grammatical functions based on their distributions, with common functions across languages including subject, object, indirect object, and others.

The next major linguistic insight into syntactic relationships, building upon the notion of grammatical function, is that of *syntactic valency*: predicates, or lexemes whose semantics consist of a functor that requires specific arguments, impose those grammatical functions that are expected by their semantic arguments. For example, the French verb predicate 'obéir' ("to obey") requires semantically a protagonist and a norm (in this case a thing that is obeyed, either an abstract or animate entity), and these are realized through the grammatical functions of subject and indirect object, respectively, as in "Pierre obéit à la loi" ("Pierre obeys the law"). There is a large area of study in linguistics called *linking theory* that is concerned with the regularities between the properties of semantic arguments and the grammatical functions in which they are realized. For instance, the seminal work of Dowty (1991) proposes the argument selection principle, which defines a set of proto-agent properties and a set of proto-patient properties; a given semantic argument can bear properties from both sets, and an argument that bears a majority of proto-agent (resp. proto-patient) properties will be lexicalized as the subject (resp. object).

1.2 Formal Representations of Syntax

Having presented the key concepts and terms concerning morphology, word forms, and basic syntactic notions in the previous section, the big issue that we will discuss here is how to formally represent syntactic grammatical structure, which must describe relationships between words in a sentence while accounting for the recursive

nature of these relationships. Turning to the formal definition of the structure of syntactic requirements between word forms, historically there are two major schools in linguistics when approaching the characterization of grammars for natural language syntax. One is *phrase-structure grammar*, a theory appearing in the 20th century and having a major influence on modern western linguistics; this characterization is based on the notion of phrases, or subsequences of word forms within a sentence that provide internal structure, and typically uses phrase generation rules as a way to model grammatical constraints. The other is *dependency grammar*, which has a historically rich tradition dating back to antiquity, and which uses a more direct approach by characterizing word form relationships in a sentence as dependencies between pairs of word forms.

We will now provide formal descriptions of the two major approaches to grammatical structure, defining the notions of *phrase* and *dependency* in the process, and then discuss under what conditions the corresponding structural representations are formally equivalent as well as why we ultimately choose to use dependency grammar in our work.

1.2.1 Phrase-Structure Grammar

The first major approach to modeling grammatical structure is termed phrase-structure grammar, and it has two key components. The most important is the definition of *phrase* units, which encompass sub-sequences of word forms in a sentence and can be recursive in structure. The second component found in most linguistic frameworks that include phrase-structure is the use of a *formal grammar* (not to be confused with the broader notion of grammar that we have discussed previously), which is a formal device used to generate sequences of terminal symbols (word forms) from re-write rules that can contain a mix of terminals and non-terminals (phrases).

Phrase-structure grammar has held a relatively prominent position in western modern linguistics as well as in the field of NLP, with the seminal work of Chomsky (1957) having a large influence in establishing formal grammars as a primary backbone of phrase-structure grammar theories. Other more recent theoretical approaches include Head-Driven Phrase Structure Grammar (Pollard and Sag, 1994) and Lexical Function Grammar (Kaplan and Bresnan, 1982), which combine formal grammars with additional, more complex linguistic apparatuses. A widespread approach in NLP is to include little linguistic overhead and rely simply on a formal grammar, with *Context-Free Grammar (CFG)* (Chomsky, 1957) as the formalism of choice; this type of grammar is noteworthy because the restrictions placed on re-write rules in CFG allow for efficient derivation algorithms while retaining sufficient expressivity to capture most syntactic constraints in languages like English

or French. Due to the widespread use and simplicity of the basic CFG approach, it will be the focus of our discussion in this section.

Phrases

Before turning to the formal generative grammar approaches to modeling phrase-structure syntax, we need to pin down the tricky linguistic notion of what constitutes a phrase. We define a *phrase* as a sequence that contains word forms and/or other phrases and that exhibits a certain cohesion supportable by linguistic tests of constituency: a phrase should be able to operate as a single unit by being primarily replaceable with a single word, and in addition translatable to different parts of a sentence and removable from a sentence, among other tests. Note, however, that no individual test for constituency is necessarily infallible, so the identification of different phrase types is generally agreed upon by looking at the entire body of linguistic evidence. It is also important to note the recursiveness built into the definition of a phrase, which as mentioned earlier is a crucial component of syntax.

Since phrases are essentially new linguistic units that are simply one level higher than word forms, it is useful to organize them, as we did with word forms, into *phrase categories* based on their syntactic distributions. Recalling that a requirement of phrases is that they be replaceable with a single word, it naturally follows that phrase category is generally defined based on the POS of the *head* of the phrase, or the one constituent word form that most influences the distribution of the phrase. Given that it is sometimes difficult to settle on a single such word, this gives room for descriptive variation: for instance, the notion of noun phrase is widely used in linguistics, but some favor the notion of determiner phrase by arguing that determiners should be considered heads.

Though specific phrase categories depend on the particular language and linguistic theoretical approach used, there is traditionally a top-level phrase for a complete sentence, as well as phrases corresponding to the open POS categories — noun, verb, adjective, and adverb phrases – as well as some of the closed grammatical POS categories – prepositional phrases, etc. We again note the usefulness of the recursive definition for phrases: we can label “le chat de la sœur de Jean” (“the cat of the sister of Jean”) as a noun phrase and account for the fact that it contains the smaller noun phrase “la sœur de Jean”, which in turn contains the smaller noun phrase “Jean”.

Formal Generative Grammar

Now that we have defined the notion of a phrase, we can present the formal rules for phrase-structure grammar. Formal phrase-structure grammars for languages traditionally follow the seminal work of Chomsky (1957), who defined several grammar families that have different restrictions on their generative rules and consequently different levels of expressive power.

In a formal grammar, a language is defined using a set of rules for rewriting strings of *symbols*, which can be either terminal or nonterminal, with terminal symbols corresponding to the language's alphabet. A string of terminal symbols is considered part of the language if and only if it can be generated using the rewrite rules of the language's grammar. The following is a detailed list of the components of a formal grammar G:

- A finite set N of *nonterminal symbols* not appearing in strings formed by G;
- A finite set Σ of *terminal symbols* that is disjoint from N;
- A finite set P of *production rules*, each rule mapping from a string of symbols containing at least one nonterminal to a new string with any content (even empty);
- A symbol $S \in N$ that is the *start symbol* from which all derivations begin.

An example of a subset of production rules capable of generating the French noun phrase "le petit chat" ("the litte cat") would be the following, with NP representing a noun phrase, N representing a noun, DET representing a determiner, and ADJ representing an adjective:

$$\begin{aligned} NP &\rightarrow DET \;\; ADJ \;\; N \\ DET &\rightarrow le \\ ADJ &\rightarrow petit \\ N &\rightarrow chat \end{aligned} \tag{1.1}$$

Moving on to the operation of a formal grammar, it can be defined in terms of relations on strings. Given a grammar $G = (N, \Sigma, P, S)$, the binary relation $x \Rightarrow_G y$ exists between a pair of strings x and y if one rule in G can be used to derive a substring in y from a corresponding substring in x in one step, or through a single rule derivation. An example from our subset of production rules above would be $[DET \;\; ADJ \;\; N \Rightarrow_G DET \;\; petit \;\; N]$. The transitive closure of this relation is $x \Rightarrow_G^*$

y, which exists between x and y if the rules in G can be used to derive a substring in y from a corresponding substring in x in zero or more steps, or through a sequence of rule derivations. An example would be $[DET \;\; NP \Rightarrow^*_G DET \;\; petit \;\; chat]$. From these operations, we can provide a definition for a valid sentence in the language as being a string of nonterminal symbols w such that $S \Rightarrow^*_G w$, meaning that it can be derived in any number of steps from the start symbol. The language of G, or $L(G)$, can then be defined as all such derivable strings from the start symbol.

Note that while our definition of formal grammars allows for arbitrary strings (as long as they contain at least one nonterminal symbol) on the left-hand side of production rules, each of our example production rules has a left-hand side consisting of exactly one nonterminal symbol. If this restriction is present in all the production rules of a grammar, then it is termed a context-free grammar (CFG). Though a detailed discussion of the expressivity of different families of formal grammars with respect to natural language syntax is outside of the scope of our work, we noted earlier in our discussion that CFG is generally agreed upon as being powerful enough to cover most natural language syntax while being simple enough to be treated efficiently using NLP algorithms. CFG is therefore the type of phrase-structure grammar that we will work with in the subsequent parts of this chapter.

Phrase-Structure with CFG

As can already be seen in our choice of symbols for the example production rules above, linguistic phrase-structure can be represented in a straight-forward manner using formal CFG grammars. Phrases correspond to non-terminal symbols, word forms correspond to terminal symbols, and POS categories, which are generalizations of word forms but not quite phrases, are represented using a restricted type of nonterminal symbol called a *preterminal*, which is defined as a nonterminal that can only participate on the left-hand side of a production rule if it rewrites to a single terminal symbol.

As for the representation of a CFG derivation, it is useful to represent the resulting syntactic structure for a sentence using a *directed graph*. In a formal mathematical sense, a directed graph consists of a set of vertices and a set of directed edges that link pairs of vertices together. For phrase-structure grammar, vertices correspond to word forms or phrases and directed edges represent the generative process, with a phrase vertex linked to its generated sub phrase and word form vertices. One of the important properties of CFG derivations is that they can be represented using *trees*, or graphs that are both connected, meaning that an edge path exists between every pair of vertices, and acyclic, meaning that no edge paths exist that leave from and return to the same vertex without traversing the same edge

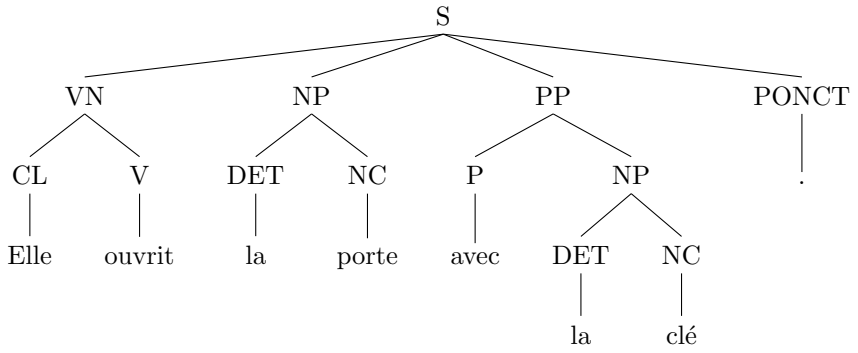


Figure 1.1: A phrase-structure tree for the sentence: "Elle ouvrit la porte avec la clé." ("She opened the door with the key.")

twice. In addition, trees derived using context-free grammars are *projective*, meaning that a derived sentence is such that every vertex dominates a contiguous substring; in other words, projective syntactic trees cannot have crossing branches. Figure 1.1 shows an example phrase-structure analysis for a sentence with CFG derivation, following the annotation scheme of the French Treebank (see Section 1.3.1).

It is important to note that some aspects of phrase-structure grammar, considering grammar in the larger sense, are not really captured by formal CFG grammars. Formal grammars make no mention of phrase heads, so syntactic rules concerning the identification of phrase heads need to be defined independently. Similarly, the syntactic notions of grammatical function and valency need to be represented. One solution has been to infer functions from the phrase-structure topology (e.g. in English, the first NP within a VP is the direct object), but this does not hold for every grammatical function and a fortiori neither for every language. So another more explicit and more general solution is to incorporate functional annotation into the phrase categories themselves: phrase categories can be subdivided into more specific ones that describe their function with respect to the head of their parent phrase, with for instance an NP-OBJ phrase indicating that the NP is an object of the parent VP's head. It should be noted, however, that annotating phrases in this manner creates only a partial functional annotation. An XP-FUNCT indicates that the XP is the FUNCT of some other item, which has to be inferred — while the inference is generally that the other item is the local head, this can be false when encountering long-distance linguistic phenomena like extraction.

As we mentioned before, more complex theories of phrase-structure grammar include additional specialized linguistic apparatuses in order to handle heads, functions, valency, etc. In Lexical Functional Grammar (Kaplan and Bresnan, 1982), two main structures work in parallel to represent the syntax of a sentence: a c-structure

representing syntactic constituents or phrases, and an f-structure representing grammatical functions. And in Head-Driven Phrase Structure Grammar (Pollard and Sag, 1994), a lexical, non-derivational approach is used to account for the syntactic requirements of grammatical function and valency.

1.2.2 Dependency Grammar

Dependency grammar, while less prominent in western modern linguists than phrase-structure grammar, became increasingly studied and popularized in the latter half of the twentieth century. It is important to note that there is, in particular, a strong linguistic tradition of using dependency syntax for slavic languages, as is the case of Czech in the Prague school of linguistics (Sgall et al., 1986) or Russian in the work of Melčuk (1988). Phrase-structure representation is problematic for these languages as they have freer word order, making it difficult to delimit phrase boundaries and infer grammatical functions from the phrase-structure topology. Dependency syntax has nonetheless also been studied for languages with stricter word order, with seminal works by Tesniere (1959) for French and Melčuk (1988) for English.

Formally, dependency grammars call for direct legitimating relationships between pairs of words, bypassing the need for definitions of phrase categories and criteria for constituency. We define these relations as follows: A word form w_g is the *governor* of w_d — and, conversely, w_d is a *dependent* of w_g — if the presence of w_g legitimates the presence and relative position of w_d (Kahane, 2001). We note as well that through transitivity of governance w_g can be said to also legitimate the presence of the dependents of w_d, and their dependents, etc.

Recalling that the identification of phrase heads in phrase-structure grammar was a bit complicated and difficult to agree upon in a number of cases, we note that similar complications arise in dependency grammar when trying to determining the direction and existence of certain legitimizing relationships. Revisiting the example of an indirect object introduced by a preposition, we note that "au cinéma" in the French sentence "il va au cinéma" ("he goes to the cinema") could potentially be represented with 'au' governing 'cinéma', or vice versa. A similar notion to that of co-heads for phrase-structure grammars can be used here, with one solution suggested by Kahane (1997) that groups co-heads into bubbles that act as single units for the purposes of dependency and governance. While this approach is perhaps better suited to representing certain linguistic phenomena, with coordination being a typical example often cited in the literature, there is the downside of a loss of generality and simplicity of the representational structure. For our purposes, bubbles are not used and order of governor is determined on a case by case basis for different constructions.

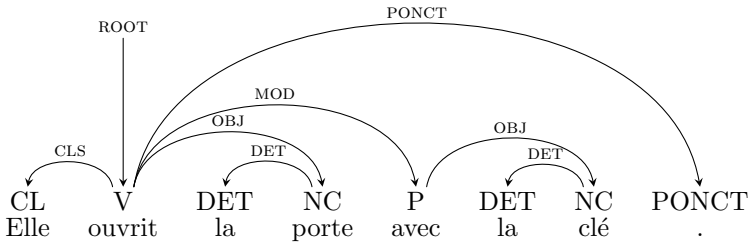


Figure 1.2: A labeled dependency tree for the sentence: "Elle ouvrit la porte avec la clé." ("She opened the door with the key.")

This leads us to fact that dependency structure can be represented, just as was the case for CFG phrase-structure, using trees. Recall that for phrase-structure grammar, vertices correspond to word forms or phrases and directed edges represent the generative process, with a phrase vertex linked to its generated sub phrase and word form vertices. For dependency grammar, on the other hand, vertices correspond to word forms and directed edges represent direct binary dependencies. A key aspect of dependency structure is the easy integration of grammatical function, through the use of what are termed *functional role labels (FRL)* that mark each directed edge in a dependency graph. A dependency tree with functional role labels is termed a *labeled dependency tree*, and we will work exclusively with such trees in this thesis. Having introduced the basic notions regarding syntactic dependency structure, Figure 1.2 shows an example dependency analysis for the French sentence "Elle ouvrit la porte avec la clé" ("She opened the door with the key") with POS categories and FRLs included.

For phrase-structure grammar we focused on the use of a mathematically formal, derivational approach to modeling syntax, but such an approach is not predominantly used for dependency grammar. Historically, many of the seminal works on dependency syntax (Tesniere, 1959; Melčuk, 1988) do not deal with formal grammars, favoring instead prescriptive rules that dictate which types of dependencies are acceptable under different circumstances and what properties the final syntactic graph structure of a sentence should have. While our focus in this thesis will likewise be limited to non-derivational approaches to dependency grammar, it is important to note that much interesting work has been done with formal grammars for dependency syntax. The use of generative dependency grammars was investigated from the very beginning of modern linguistic theories of dependency syntax (Hays, 1964; Gaifman, 1965). And more recently, in the last decade of NLP research, there have been efforts (Nasr, 2004) to apply generative dependency grammars to the problem of syntactic parsing (see Kübler et al., 2009, for an overview).

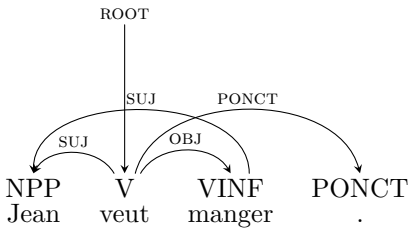


Figure 1.3: A labeled non-tree dependency graph for the sentence: "Jean veut manger." ("Jean wants to eat.")

Characteristics of Dependency Syntax

We present here dependency grammar from the traditional perspective, which is to say using prescriptive rules determining the structural nature of a dependency graph for a sentence. The following conditions, generally agreed-upon in the literature, apply to dependencies:

- *binary*, between two word forms;
- *directed*, so that one w_g is *governor* while the other w_d is *dependent*;
- *anti-reflexive*, with no word form serving as its own governor;
- *labeled* in a way that distinguishes its grammatical function.

One upside of eschewing formal grammars, with an approach to dependency syntax that is less formally structured, is that dependency grammar is very flexible and can represent a variety of linguistic phenomena that are not easily representable using trees. One example can be seen in the difference between what are called *surface* and *deep* syntactic dependencies. For certain verb predicate constructions, one verb governs a subject on the surface, while from a deeper point of view one might identify additional verbs as governor. This can typically be observed with control verbs, such as the verb 'pouvoir' in the French sentence "Jean peut manger" ("Jean can eat"). We would like the verb 'manger' to have 'Jean' as a subject, with a resulting non-tree structure as shown in Figure 1.3. This is because 'manger' imposes semantic selectional restrictions on that position, as can be seen with the sentence "# La lune peut manger" ("The moon can eat"), which is not acceptable because of the incompatibility of 'lune' as a subject of 'manger'. A downside to allowing multiple governors for a single word form, however, is that the syntactic analysis becomes much more complicated by no longer conforming to a tree structure.

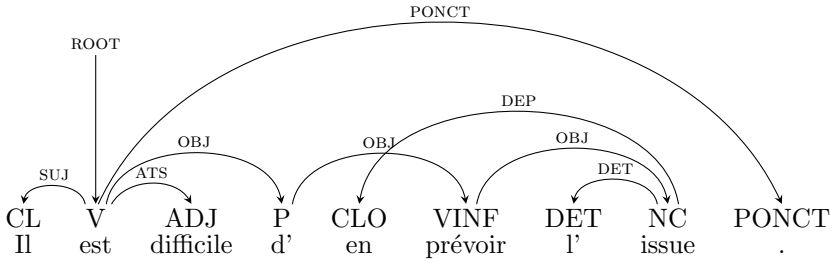


Figure 1.4: A labeled non-projective dependency tree for the sentence: "Il est difficile d'en prévoir l'issue." ("It is difficult to foresee the result [of it].")

Another phenomenon that dependency structure is capable of representing is that of sentences for which tree representations of their dependencies must be non-projective. This is true for French in cases of unbounded extraction, as well as, for instance, the pronominalization of a 'de' PP that depends on a direct object, as noted in a recap of non-projectivity in French text corpora by (Candito and Seddah, 2012b). An example of the latter case can be seen in the sentence "Il est difficile d'en prévoir l'issue" ("It is difficult to foresee the result [of it]"). Figure 1.4 shows a non-projective dependency tree analysis for this sentence, with intersecting arcs representing the violation of tree projectivity.

Granted that dependency structure can be used to capture interesting non-projective or non-tree phenomena, it is nonetheless useful to consider the restriction of dependency syntax to projective trees. If we are interested in representing more straightforward surface structure, the projective tree requirement is linguistically viable for French in most cases; as Candito et al. (2010a) note, non-projectivity occurs empirically quite rarely in French text. One interesting representational problem that remains is the treatment of coordination in dependency syntax, with a good discussion of this issue presented by Kahane (2001); we will hold off discussing this for now and return to it in our discussion of French syntax and resources. Ultimately, in this thesis we use the simplifying projective tree structure assumption for representing dependency syntax, meaning that the dependency graph for a sentence ends up being a tree and having the following additional properties, adapted from Melčuk (1988):

- *connectedness*, which prohibits any detached subparts;
- *acyclicity*, which prohibits any directed dependency cycles in the graph;
- *unique governance*, which prohibits nodes from having more than one incoming arc (i.e. no word form may have more than one governor);

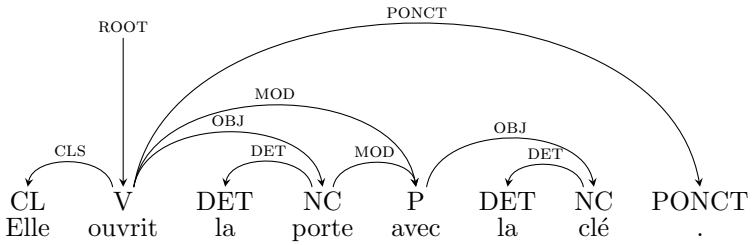


Figure 1.5: Illustration of artificial syntactic ambiguity, with the arc from 'ouvrit' to 'avec' for true structure and the arc from 'porte' to 'avec' for another grammatically licit structure, in a labeled dependency analysis for the sentence: "Elle ouvrit la porte avec la clé." ("She opened the door with the key.")

- *root governance*, which requires that a single *root* node have no governor.

1.2.3 Syntactic Ambiguity

Earlier, we distinguished between grammaticality and acceptability as a decoupling of a sentence's validity according to, respectively, the language's grammar and the sentence's interpretability by a speaker of the language. Having described syntactic structure for both major grammatical formalisms, we can now illustrate the problem of *syntactic ambiguity*, which occurs when an acceptable sentence has multiple interpretations, each with a different corresponding syntactic analysis. This issue arises regardless of the choice of grammatical formalism, though for simplicity we will use dependency syntax in the following examples.

One type of ambiguity at the syntactic level can be termed *artificial syntactic ambiguity*, with multiple syntactic structures for a sentence being licit according to the grammar but only one true structure corresponding to its correct semantic interpretation. As an example, consider the French sentence "Elle ouvrit la porte avec la clé" ("She opened the door with the key"). In the correct interpretation the key is used to carry out the act of opening, while in the highly suspect interpretation the key is a descriptor for the door. Both interpretations, however, have grammatically valid syntactic analyses, as shown in Figure 1.5 with the arc from 'ouvrit' to 'avec' corresponding to the structure with correct interpretation and the arc from 'porte' to 'avec' corresponding to the grammatically licit structure with incorrect interpretation. This type of ambiguity is particularly relevant to syntactic parsing, which is given a sentence and tasked with finding its correct syntactic structure; it is not sufficient to find a structure that is licit according to the grammar, and for this reason syntactic parsing is a semantic task in addition to a syntactic one.

Alternatively, we term *true syntactic ambiguity* the case where it is unclear

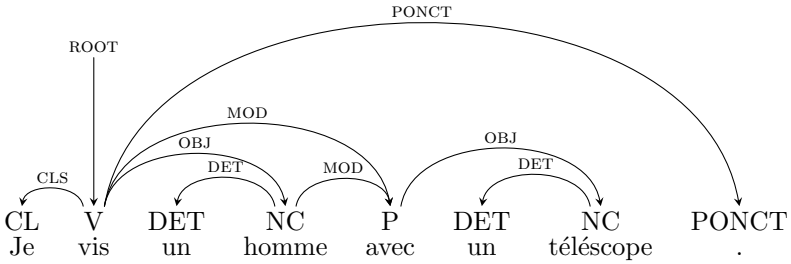


Figure 1.6: Illustration of true syntactic ambiguity, with arcs from 'vis' to 'avec' and from 'homme' to 'avec' indicating competing licit grammatical structures, in a labeled dependency analysis for the sentence: "Je vis un homme avec un telescope." ("I saw a man with a telescope.")

which interpretation is correct from the available context, resulting in multiple syntactic analyses that have acceptable interpretations. An example is the French sentence "Je vis un homme avec un telescope" ("I saw a man with a telescope"). In one interpretation the telescope is used to carry out the act of viewing, while in the other interpretation the telescope is a descriptor for the man (i.e. he is carrying it) — more context is needed to identify the correct interpretation. Both interpretations are represented in dependency structure in Figure 1.6, with arcs from 'vis' to 'avec' and from 'homme' to 'avec' indicating competing analyses.

1.2.4 Formalism Equivalence

An interesting fact concerning the relationship between the two types of grammars introduced here is that headed phrase-structure CFG trees and projective dependency trees are actually formally equivalent (Robinson, 1970). A recursive head propagation procedure, popularized by Magerman (1995) and later Yamada and Matsumoto (2003) for the purpose of creating data for dependency parsing, can be used to convert any headed phrase-structure CFG tree into a corresponding projective dependency tree. Below, we provide a formal description of the process of converting from phrase-structure representation to dependency representation.

Conversion Process

The conversion process takes as input a headed phrase-structure tree with nonterminal symbol vertices V_n, terminal symbol vertices V_t, and directed edges E_{ps}, giving a graph $(V_n \cup V_t, E_{ps})$. The output is a dependency tree that retains only the terminal symbol vertices V_t, along with a new top level v_{ROOT} vertex, and creates a new set of directed edges E_d, giving a graph $(V_t \cup \{v_{\text{ROOT}}\}, E_d)$. One can produce either labeled

or unlabeled output dependency trees, with labeling achieved using heuristics and functional annotation of phrases (cf. Section 1.2.1). We assume a labeled conversion here, though we will not go into details concerning the heuristics; we leave that to a later discussion of the conversion process for the French resource we use.

During conversion, a recursive function R is called on subtrees at non-terminal vertices v, beginning with the start symbol vertex v_s that represents the full tree. The function $R(v)$ returns the terminal head vertex of v, adding relevant edges to the new dependency tree in the process. $R(v)$ is presented formally below, with the (v_1, v_2) representing a directed edge from a vertex v_1 to a vertex v_2, and (v_1, l, v_2) for labeled dependencies with label l:

1. Identify the children vertices C of v. Formally, $C \leftarrow \{x : (v, x) \in E_{ps}\}$.

2. For any $x \in C$ such that x is a nonterminal vertex:

   - Recursively find the terminal head h_x of x. Formally, $h_x \leftarrow R(x)$.
   - Remove x from C and add h_x to C. Formally, $C \leftarrow (C - \{x\}) + \{h_x\}$.
   - If x was a phrase with a functional annotation a, set $f(h_x) \leftarrow a$.

3. Identify the head child h_v of v.

4. For each $h_x \in C - \{h_v\}$:

   - Let label l_x be $f(h_x)$ if it exists, otherwise use a heuristic to determine l_x.
   - Add a new edge from h_v to h_x. Formally, $E_d \leftarrow E_d + \{(h_v, l_x, h_x)\}$.

5. Return h_v.

The conversion process can then be carried out by simply calling the function $R(v_s)$. A last step after the recursive process is to add a final edge $(v_{\text{ROOT}}, \text{ROOT}, h_s)$ in the converted dependency tree from a dummy v_{ROOT} vertex to the head vertex of the sentence h_s with dummy label ROOT. It should be noted that the resulting dependency tree is necessarily projective, due to the projective nature of the original phrase-structure, meaning that the representation of non-projective phenomena would necessitate a post-processing step to find and modify the dependency graph.

Advantages of Dependency Syntax

For this thesis, we have chosen to focus our efforts on the investigation of syntactic parsing under a single formalism. The important question arises, then, of which of these two major formalisms to use?

On one hand, we have some practical considerations to take into account. Looking ahead for a moment, one such consideration is the availability of syntactic treebank resources, which contain syntactic analyses of large numbers of sentences in a language and are necessary for training accurate computational models for parsing. We note that many languages, including English and French, have resources built from large-scale syntactic annotation efforts using phrase-structure syntax. However, as dependency syntax has become more popular in the area of NLP, many of these resources have been converted into dependency structure, using recursive procedures like the one described above. As a result we are not obligated to choose one formalism or the other based on resource availability. Another practical consideration is the computational efficiency of *parsing* algorithms, which automatically predict the syntactic structure of a sentence. Algorithms for generative CFG phrase-structure parsing take at least cubic time in the length of the sentence, as is the case for the widely-used CKY (Kasami, 1965; Younger, 1967; Cocke and Schwartz, 1969) and Early (Earley, 1970) algorithms, while dependency parsing can achieve linear time efficiency using transition-based algorithms (Nivre, 2003; Yamada and Matsumoto, 2003) that we will discuss in more detail in Chapter 2.

There are also linguistic considerations to take into account. From a linguistic perspective our preference is also for dependency syntax, because it is simpler and makes less assumptions about intermediate levels of syntactic structure. As Melčuk (1988) notes, phrase-structure syntax arose primarily as a way to model English, a language with very restricted word order that fits well into the constituency paradigm, yet other languages, such as the Slavic languages studied by both Tesnière and Melcuk, do not lend themselves well to this representation. Given the relative simplicity and fewer assumptions made by dependency syntax, one can argue that it is more appropriate for research on syntactic parsing, especially considering that global interest in parsing is increasing and computational models should ideally be applicable to many languages. Another linguistic argument in favor of dependency syntax is that dependencies encode grammatical relations in a more complete manner, which facilitates the passage to argument structures between predicates and their semantic arguments. If the purpose of syntax is to study the way in which the form of a sentence organizes itself to carry meaning from one speaker to another, then it is advantageous to use a syntactic formalism that does not use more structure than is necessary to communicate meaning.

Given the above considerations, we have decided in this thesis to use exclusively the dependency syntax formalism in our syntactic parsing research. For French, the relatively recent creation of a resource containing dependency analyses of sentences followed the trend of conversion from phrase-structure resources described above. In the next section, we will thus discuss both the original phrase-structure resource and its subsequent conversion into dependency syntax, the latter of which constitutes the core data set for our parsing experiments.

1.3 French Syntax and Resources

For the purposes of this thesis, a discussion of French syntax and a discussion of the syntactic analysis resource we use for French go together hand in hand, since decisions made concerning the various elements of the resource — the set of POS categories, the set of functional role labels, and the manner in which complex syntactic phenomena are represented — reflect larger grammatical properties of the language. We thus center our discussion in this section around the resources that we use for French, with digressions when necessary into broader aspects of French syntax.

In recent decades, there have been a number of concerted efforts to build human-annotated syntactic treebanks for various natural languages, with the goal of aiding empirical linguistic study and spurring the development of computational models of syntax. *Treebanks* are defined as resources in which sentences taken from a corpus of natural language text are annotated with corresponding syntactic representations adhering to a particular grammatical formalism. Major efforts started as far back as the early 1990's with the Penn Treebank (PTB) (Marcus et al., 1993) for English, and the French Treebank (FTB) (Abeillé et al., 2003) was released in the early 2000's.

Individual treebanks across different languages vary according to a number of criteria, with the most important being the fundamental choice of syntactic representation, either phrase-structure syntax or dependency syntax. There have been efforts across languages at manually annotating treebanks with both phrase-structure, as is the case for the PTB and FTB, and with dependencies, as is the case with for instance the Prague Dependency Treebank (Böhmová et al., 2003). Another option that has been explored is the creation of dependency treebanks by conversion from a phrase-structure treebank in the same language, as has been done a number of times for English (Magerman, 1994; Yamada and Matsumoto, 2003; Johansson and Nugues, 2007). This option was used to create the *French Dependency Treebank (FTBDep)* (Candito et al., 2010a), which is the primary treebank resource used

in our thesis, with a conversion process starting from the original phrase-structure FTB.

We will now describe the original phrase-structure FTB and the decisions made concerning its annotation. This will be followed by a presentation of the converted dependency FTBDep, whose representational characteristics are determined primarily by those of the original FTB but additionally by head identification choices, as well as modifications to the phrase-structure representation of certain linguistic constructions prior to conversion.

1.3.1 The (Phrase-Structure) French Treebank

The FTB (Abeillé et al., 2003) is a resource built and enriched at the Université Paris VII, in the TALANA lab. This resource is considered the definitive large-scale syntactically annotated resource available for French, and was created from extracts of the *Le Monde* French newspaper, ranging from the years 1989 to 1993, with representative text from different domains such as economy, literature, and politics. It contains 20,000 sentences manually annotated with phrase-structure syntax, and as of 2007 it had functional annotations for 12,351 of those sentences. Within the portion of the treebank with functional annotations, this amounts to 370,000 words (before considering the identification and merging of compound words).

Manual annotation of the treebank is divided into two levels: the morphosyntactic level identifies important information at the level of word forms, while the phrase-structure level is built on top of the morphosyntactic annotations and provides a phrase-structure tree with functional annotations for each sentence. We discuss the tag sets and other representational choices below.

Morphosyntactic Annotation

The morphosyntactic annotation in the FTB identifies a variety of information at the word form level: POS categories, some sub-POS categories (as in possessive, cardinal, etc.), inflection, lemmatization, and compound identification. Table 1.2 includes the annotation tagsets used, with sub-POS and inflection possibilities listed for each POS category.

The set of POS categories in the FTB is mostly traditional, with widely accepted choices for the major categories. One interesting choice is the use of a separate POS category for *clitics*, or weak pronouns, which follows the generative tradition Kayne (1975). Additionally, foreign words receive a special POS, punctuation marks are treated as lexemes and assigned a special POS, and most typographical signs like numbers and abbreviations are assigned to an appropriate POS (e.g. in “7 janvier”,

POS	Sub-POS	Inflection	Description
A	card, ord, poss, qualif, indef, inter	gender+number+person	Adjectives
ADV	-, inter, exclam, neg	-	Adverbs
CL	-, subj, refl, obj	gender+number+person	Clitics
C	subord, coord	-	Conjunctions
D	card, dem, def, indef, exclam, neg, poss, inter, part	gender+number+person	Determiners
ET	-	-	Foreign words
I	-	-	Interjections
N	common, proper	gender+number	Nouns
P	-	-	Prepositions
PRO	inter, pers, card, neg, poss, rel, indef	gender+number+person	Pronouns
PONCT	strong, weak	-	Punctuation
PREF	-	-	Prefixes
V	-	gender+number+person +mood+tense	Verbs

Table 1.2: Tagsets and markers at the morphosyntactic level of annotation in the FTB.

the '7' is an adjective). As explained in Section 1.1.2, the distributional nature of the organization of lexemes into POS categories means that such categories can be defined at different granularities, with different amounts of distributional overlap among lexemes in a category. In the FTB, finer distinctions within the traditional, coarse POS categories are annotated, distinguishing between proper and common nouns, subordinating and coordinating conjunctions, among others.

The next set of annotations at the morphosyntactic level concerns inflection. The marking of inflection is important, especially for French, because an important syntactic requirement for phrases is inflectional agreement (e.g. between subject and verb), and additionally because many word forms in French are ambiguous with respect to mood, person, number, and gender. As Abeillé et al. (2003) indicate, the French verb form 'mange' ('eat') can be either indicative or subjunctive, either first or third person, and can even be the second person imperative.

A subsequent annotation step is the identification of lemmas, which is useful due to ambiguity between homographic word forms even after POS categories have been identified. For example, when considering the French word form 'être', it is ambiguous between the verb 'to be' and the common noun for a 'being' or 'entity'. In addition, when considering the French word form 'suis', even after identifying it as a verb it is still ambiguous between the first person present indicative singular forms of 'être' and 'suivre' ('to follow'). In addition, lemmas are useful markers when using the corpus for various purposes, from corpus queries to refined valency

annotation.

A final consideration when annotating at the morphosyntactic level is the identification of compounds. In Section 1.1.1, we defined a compound word as having a minimal meaning that cannot be properly decomposed if it were to be divided into separate word forms. In the FTB annotation, however, the criteria for compound words covers a larger number of cases. As Crabbé and Candito (2008) note, in the FTB 14.5% of tokens are part of a compound word and include sequences in which: the compound cannot exist separately, as in "aujourd'hui" ('today'); the semantics are non-compositional, as in "carte bleue" ("credit card"); the syntax is irregular, as in "à la va vite" ("in a hurry"); the compound is a fixed verbal expression, as in "mettre en garde" ('warn'); the compound is a named entity, as in "Union hospitalière privée"; the semantics are compositional but insertion cannot occur or does so rarely, as in "garde d'enfants" ("child care"). Finally, numerical amounts expressed with multiple tokens of digits or letters are also marked as compound words, accounting for around 10% of compound occurrences.

Phrase-Structure and Functional Annotation

Having finished our description of the morphosyntactic level, we now move on to the phrase-structure level. The phrase-structure annotation in the FTB identifies groupings of phrases in sentences, with a focus on surface annotations with little internal structure that are compatible with various syntactic frameworks. The following information is included at the syntactic level of annotation: main category, possible sub-category, surface function, opening or closing boundaries, and diathesis for verbal nuclei. Table 1.3 includes the phrase category and surface function annotation tagsets, with sub-phrase category and surface function possibilities listed for each phrase category.

The major phrase categories used by the FTB are, as we saw earlier for the POS categories, traditional and widely accepted linguistic choices. Abeillé et al. (2003) note that for the sake of simplicity unary phrases, or phrases with a single constituent, are used only when needed: unary noun phrases are used for proper names and pronouns but not for bare common nouns; unary adjective phrases are used for predicative adjectives but not for prenominal ones; and unary verb nucleus phrases are used for single verbs, while no unary adverb phrase is used. As a way marking certain key distinctions within the major phrase categories, the FTB annotation uses sub-phrase categories, which encode information such as relative (rel) vs. subordinate (sub) clause type for embedded S clauses.

Surface function annotation, which was carried out during a subsequent effort to enrich the FTB (Abeillé and Barrier, 2004), is a way of including grammatical

Phrase	Sub-Phrase	Surface Function	Description
(NP ...)	-	SUJ, OBJ, ATS	Noun phrases
(VN ...)	-	SUJ, A-OBJ, D-OBJ	Verbal nuclei
(VP ...)	-, inf, part	OBJ, A-OBJ, D-OBJ, P-OBJ, ATS	Infinitives and nonfinite clauses
(PP ...)	-	A-OBJ, D-OBJ, P-OBJ, ATS, ATO	Prepositional phrases
(ADVP ...)	-	-	Adverbial phrases
(AP ...)	-	ATS, ATO	Adjectival phrases
(SENT ...)	-	-	Sentences
(S ...)	-, int, sub, rel	OBJ, D-OBJ, ATS	Finite clauses
(COORD ...)	-	-	Coordinated phrases

Table 1.3: Tagsets and markers at the phrase-structure level of annotation in the FTB. The additional surface function MOD is applicable to any phrase category

function information in a phrase-structure tree. In the case of the FTB, this was applied exclusively to arguments of verbs. Following the established convention, surface functions do not mark edges in the phrase-structure tree but rather vertices represented the functional phrases themselves; for example, to represent that a noun phrase is the subject of a sentence, the noun phrase is labeled NP-SUJ.

We would like to additionally call attention to two interesting linguistic phenomena, verb phrases and coordination, for which the FTB makes specific choices regarding their representation. For verb phrases, Abeillé et al. (2003) choose to annotate the minimal verbal nucleus (VN), which includes clitics, auxiliaries, negation and verb. They argue against using the traditional notion of a verb phrase that includes objects and modifiers because the traditional verb phrase is often discontinuous in French: for instance, the subject can appear after the verb, especially in the context of extraction, and it can then intervene between the verb and other arguments. This can be seen with the intervening subject 'IBM' in the phrase "Les actions qu'a mises IBM sur le marché" ("the shares that IBM put on the market"). In the FTB annotation scheme, the VP phrase category is thus used only for infinitival and participial clauses.

For coordination, they choose to delimit coordinating phrases to include only a coordinating conjunction and its right conjunct, and they also leave coordinating phrases underspecified with respect to phrase category. Each pair of coordinating conjunction and conjunct after the first is thus annotated with the category COORD, resulting in a flat structure when coordinating between more than two conjuncts, as shown in Figure 1.7 for the sentence "Jean voit Pierre, Paul et Marie" ("Jean sees Pierre, Paul and Marie"). Note that the comma acts as a strong punctuation mark and essentially takes on the category of coordinating conjunction, while technically

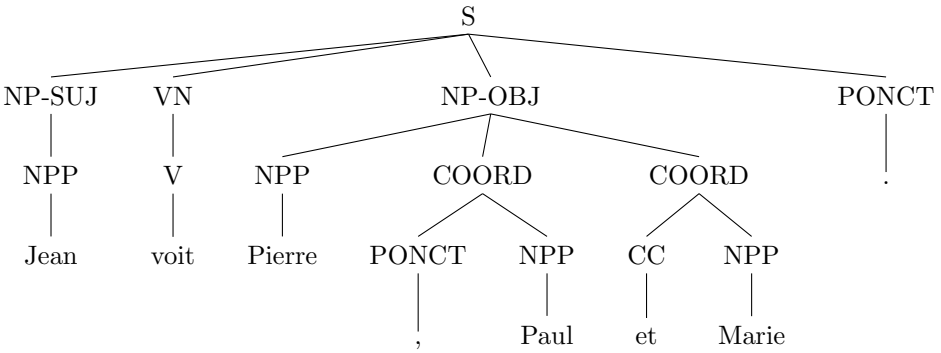


Figure 1.7: A phrase-structure tree for the sentence: "Jean voit Pierre, Paul et Marie" ("Jean sees Pierre, Paul and Marie").

retaining the punctuation POS category.

1.3.2 Conversion to the French Dependency Treebank

Now that we have presented the annotation details at the morphosyntactic and phrase-structure level for the FTB, we can continue on to a discussion of the conversion into dependencies that led to the creation of the FTBDep. This research effort took place very recently, this time carried out by the interdisciplinary research group Alpage and in particular through the work of Candito et al. (2010a). As we did for the FTB presentation, we will divide our discussion of the treebank conversion process into two parts: minor modifications of the FTB's morphosyntactic annotation, and then the major conversion effort of phrase-structure annotation into dependency annotation.

Morphosyntactic Modifications

A first modification made by Candito et al. (2010a) was the undoing of some compound words that had been annotated in the original FTB. We previously noted that Abeillé et al. (2003) mark compound words under a number of different circumstances. One key result is that in a number of cases the compounds are *syntactically regular*, meaning that when split up they still adhere to the language's syntax. Candito et al. (2010a) choose to adopt a minimal representation of compounds that undoes all the ones that are syntactical regular. They justify this choice by noting that syntactically regular compounds are often those that are most debatable to be marked as compounds; they note that as a result there are a number of annotation inconsistencies in the FTB where the same sequences are sometimes marked as compounds and sometimes not.

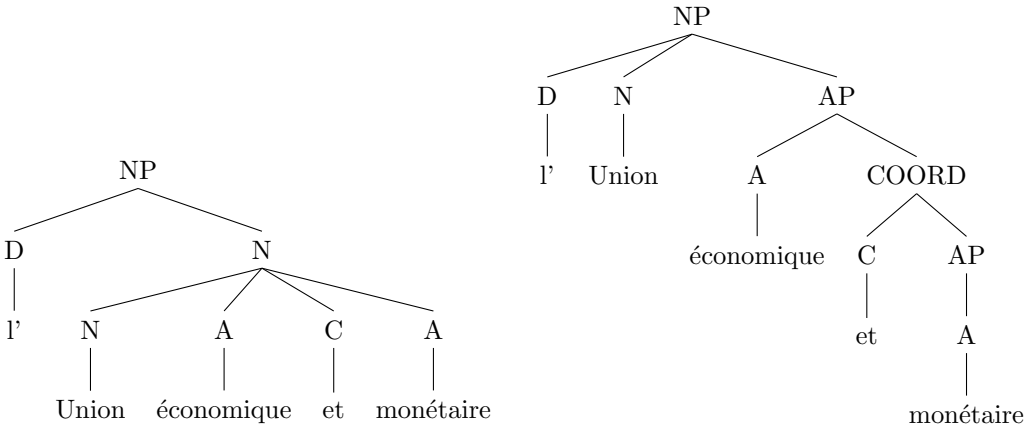


Figure 1.8: Undoing of a compound from the original FTB (left) to the FTB-UC (right) for the syntactically regular compound: "l'Union économique et monétaire" ("the economic and monetary union").

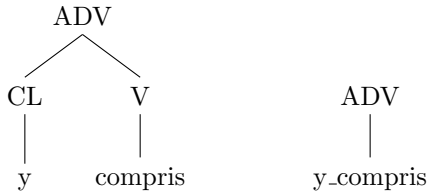


Figure 1.9: Merging of a compound from the FTB (left) to the FTB-UC (right) for the sequence: "y compris" ("including").

These modifications led to added structure in the resulting phrase structure trees, as can be seen Figure 1.8. The variant of the phrase-structure FTB after this modification is given the name FTB-UC, with the number of distinct compounds in the FTB-UC reduced to almost half of the original number. For the remaining compounds, in FTB-UC these are merged into single compound word forms retaining the POS category of the original flat phrase, as can be seen in Figure 1.9 for the compound "y compris" ("including"). Representing those syntactically regular compounds using regular phrase structure should be transparent for syntactic analysis, but it should be noted that downstream NLP applications require additional treatment to recover the potentially non compositional meanings of such sequences.

Another morphosyntactic modification prior to conversion was the use of a modified POS tagset, as presented by Crabbé and Candito (2008). Two granularities are used, with the coarse-grained categories matching up to the standard POS categories from the original FTB, while the fine-grained categories additionally make distinctions using some of the sub-POS information discussed earlier for the FTB

Coarse POS	Fine POS	Description
V	V	Indicative verb
	VS	Subjunctive verb
	VINF	Infinitive verb
	VPP	Past participle
	VPR	Present participle
	VIMP	Imperative verb
N	NC	Common noun
	NPP	Proper noun
C	CS	Subordinating conjunction
	CC	Coordinating conjunction
CL	CLS	Subject clitic
	CLO	Object clitic
	CLR	Reflexive clitic
P	P	Preposition alone
P+D	P+D	Preposition with determiner
P+PRO	P+PRO	Preposition with pronoun
I	I	Interjection
PREF	PREF	Prefix
PONCT	PONCT	Punctuation
ET	ET	Foreign word
A	ADJ	Adjective other than interrogative
	ADJWH	Interrogative adjective
ADV	ADV	Adverb other than interrogative
	ADVWH	Interrogative adverb
PRO	PRO	Pronoun other than relative or interrogative
	PROREL	Relative pronoun
	PROWH	Interrogative pronoun
DET	DET	Determiner other than interrogative
	DETWH	Interrogative determiner

Table 1.4: List of coarse-grained and fine-grained POS categories used in the French Dependency Treebank (FTBDep).

(such as common vs. proper nouns) as well as verbal mood information, wh-features, and other factors. The full list of POS categories used in the FTBDep is shown in Table 1.4.

Syntactic Conversion

Beyond the modifications to the morphosyntactic annotation of the FTB, an additional syntactic preprocessing step is taken before the recursive head-propagation procedure converts phrase-structure trees into dependency trees. In this step, a treatment of prepositions and complementizers is applied to the phrase-structure trees of the FTB that ensures their projection of PP and S phrases, respectively. These modifications are applied in order to fulfill a linguistic choice for the result-

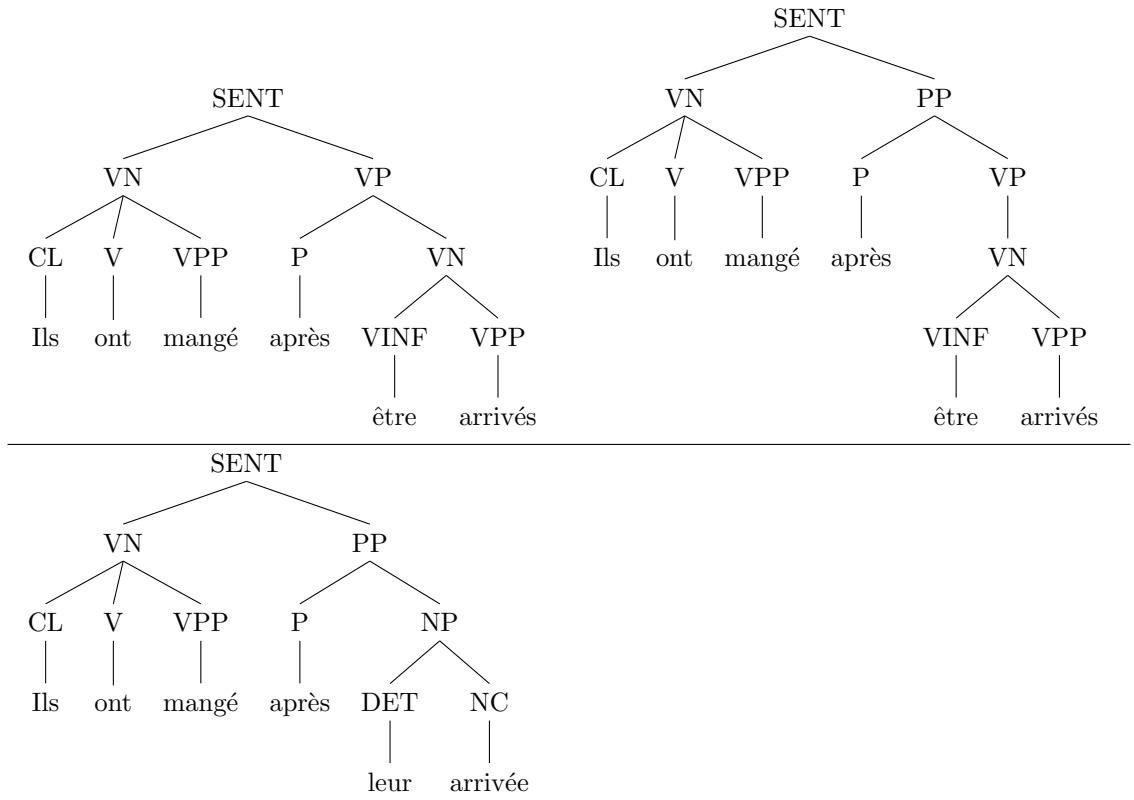


Figure 1.10: Raising of a preposition from the original FTB scheme (above left) to the FTB-UC scheme (above right) for a PP with infinitival object for the sentence: "Ils ont mangé après être arrivés" ("They ate after arriving"). Additionally, an unchanged example of a PP with nominal object (below) for the sentence: "Ils ont mangé après leur arrivée" ("They ate after their arrival").

ing dependency trees, which is that complementizers and prepositions be chosen as heads of the constituents they introduce. For PPs, this achieves a normalization of the cases with nominal PP-object and infinitival PP-object, as illustrated in Figure 1.10 for two similar sentences "Ils ont mangé après être arrivés" ("They ate after arriving") and "Ils ont mangé après leur arrivée" ("They ate after their arrival"). For S phrases, an intermediate Sint phrase is created and the complementizer is raised, as illustrated in Figure 1.11 for the sentence "Je sais que Paul aime Julie" ("I know that Paul loves Julie"). Note that this applies both when the complementizer is semantically empty, as for 'que', and when it is a meaningful conjunction, as for 'quand' ('when') in the sentence "Je partirai en vacances quand j'aurai fini ma thèse" ("I will go on vacation when I have finished my thesis").

The next step is to use the recursive head-propagation procedure described in Section 1.2.4. As a source of head-finding rules, Candito et al. (2010a) use a

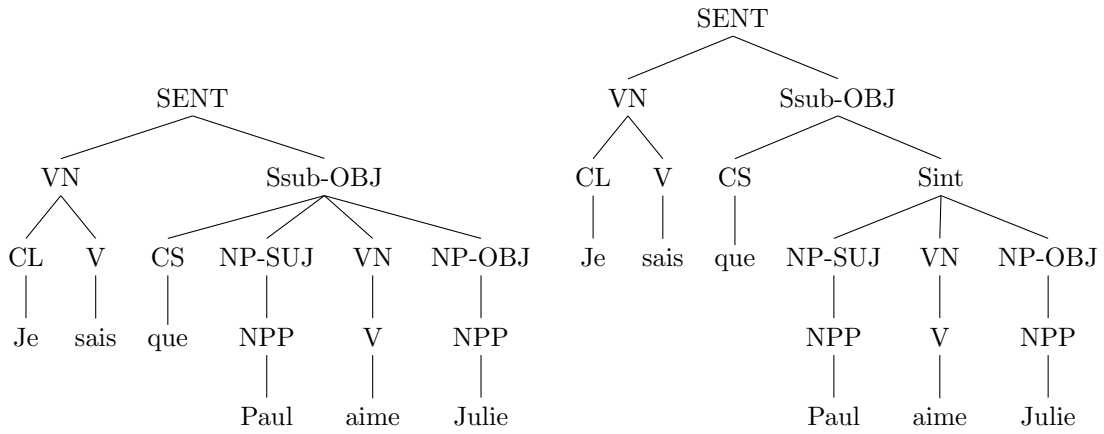


Figure 1.11: Rasing of a complementizer from the original FTB scheme (left) to the FTB-UC scheme (right) for a sentential complement in the sentence: "Je sais que Paul aime Julie" ("I know that Paul loves Julie").

modified version of the rules of Arun and Keller (2005) adapted to French and to the FTB annotation scheme. During the conversion process, Candito et al. (2010a) also make use of the surface function annotations in the FTB to add functional role labels to some of the edges in the newly created dependency trees, although note that the FTB only provides surface function annotation for subcategorized dependents of verbs. For other types of dependencies, it is often clear which grammatical function should be used, and thus heuristics are used to label edges for dependents of non-verbs, dependents of verbs that do not project a phrase, coordinated phrases, and dependents of adnominal participles. An example from Candito et al. (2010a) concerns the noun phrase "vêtements achetés par les Français" ("clothes bought by the French"), where the PP "par les Français" has no surface function in the FTB and is thus labeled using a heuristic as serving a P-OBJ of the past participle 'achetés'. The full list of functional role labels used in the final FTBDep is shown in Table 1.5.

It is clear that the annotation scheme of the original phrase-structure FTB largely dictates the linguistic choices evident in the resulting dependencies in the FTBDep. However, a number of nontrivial linguistic choices are made by the head-finding rules; as we have noted many times in this chapter, the identification of phrase heads for a number of linguistic phenomena is up for debate, and the conversion from phrase-structure into direct dependencies forces these previously underspecified decisions to be made explicit. Candito et al. (2010a) identify three important linguistic phenomena for which specific head-finding decisions were made:

1. Verb auxiliaries: Tense, passive, and causative auxiliaries are treated as de-

Surface Label	Description
SUJ	Subject
OBJ	Object
DE_OBJ	Indirect object with preposition “de” (“of”)
A_OBJ	Indirect object with preposition “à” (“at”)
P_OBJ	Indirect object with preposition other than “de” or “à”
ATS	Subject complement
ATO	Object complement
AUX_TPS	Temporal auxiliary
AUX_PASS	Passive auxiliary
AUX_CAUS	Causative auxiliary
AFF	Affix (for fixed clitics)
MOD	Modifier other than relative clause
MOD_REL	Relative clause
COORD	Coordination (for conjunctions, with first conjunct as governor)
ARG	Argument (for linked prepositions)
DEP_COORD	Coordination (for conjuncts, with previous conjunction as governor)
DET	Determiner
PONCT	Punctuation (except commas that are coordinating conjunctions)
DEP	Prepositional dependent (with non-verbal governor)

Table 1.5: List of surface functional role labels used in the French Dependency Treebank (FTBDep).

pendents of the past participle or infinitive they introduce. For example, the VN “a quitté” (“has left”) is represented in the FTB using a flat structure with ‘a’ and ‘quitté’ at the same level, while the FTBDep explicitly creates a dependency with functional role label AUX_TPS between the governor ‘quitté’ and the dependent ‘a’.

2. Prepositions and complementizers: Recalling the preprocessing step in which these were raised in the phrase-structure trees to always project PP and S phrases, respectively, during conversion these are systematically treated as the head of the complement they introduce.

3. Coordination: As we have described earlier, the original FTB uses a flat structure for coordination in which each conjunct after the first is placed in a COORD phrase along with its introducing conjunction. During conversion, the same philosophy is maintained by identifying the first conjunct as the head of the phrase.

As a final note concerning the FTBDep resource, we highlight the fact that the syntactic tree analyses in the FTBDep are subject to the projectivity constraint; due to its automatic conversion from a projective phrase-structure resource, the

FTBDep consists entirely of projective trees. However, as we indicated earlier in Section 1.2.2, there exist linguistic phenomena in natural language that necessitate non-local dependencies violating the projectivity constraint. In order to gauge the amount of representational error caused by the projectivity restriction for French, Candito et al. (2010a) performed a manual annotation of non-local dependencies for the first 120 sentences of the converted FTBDep. They found that non-projectivity occurred primarily during linguistic extraction: whether out of a sentence, out of an NP using the relative pronoun 'dont' ('of which'), or out of an NP using the clitic pronoun 'en' ('of it'). Importantly, the non-projective edges comprised only 1.22% of the total number of edges in their sample. Due to the complications involved with correcting the non-projective edges throughout the entire FTBDep, as well as the fact that non-projective edges occur rarely in French, we willingly accept the projective constraint imposed by our data and this constraint will be implicitly present throughout the rest of this thesis.

1.4 Machine Learning Methods

At its heart, the focus of our work is to build statistical models that are able to accurately parse and correct the syntactic structure of sentences. In this section, we will provide an introduction to the most widely-used approaches for obtaining these models, which fall under the umbrella term known as *machine learning*, and then focus on those techniques that we use in this thesis. It is important to note that we do not intend for this to be a comprehensive introduction to machine learning methods for a reader with little or no prior knowledge. Rather, it is meant to explicitly define the algorithms and techniques we choose to use, with theory and mathematical formulas included as a matter of course.

The field of machine learning has long been a rich source of algorithms and techniques that are applicable to tasks in NLP such as POS tagging, parsing, and others. In this thesis, we employ machine learning algorithms that are compatible with the classic *supervised learning* paradigm, which essentially learns from example. A set of training examples is taken as input, with each example containing *features*, or informational clues, that are hopefully sufficient for determining the value of a corresponding label. From this training set, the goal is to induce a model capable of looking at novel examples of the same kind and correctly predicting their labels from the features.

We are particularly interested in algorithms for supervised learning that use linear models. This subclass of models has the downside of assuming statistical independence between different features; for example, if we are trying to predict whether

it will rain today and our two feature variables are whether it rained yesterday and whether it rained the day before yesterday, it is clear that the two variables are not independent. On the other hand, the upside of linear models is that they allow for very efficient representations and fast training and prediction time, which is important when a learning problem calls for a large number of training examples and/or features.

Formally, a supervised learning problem with linear modeling can be described as follows:

1. A *training set* T consisting of a sequence of example tuples of the form $(\boldsymbol{x}_i, y_i)$, where $\boldsymbol{x}_i \in \mathbb{R}^n$ is a real-valued instance vector in a feature space of dimension n, and $y_i \in \{-1, +1\}$ is a binary label;

2. A *linear model* L that predicts a label $\hat{y}$ for an input instance $\boldsymbol{x} \in \mathbb{R}^k$ using a decision value calculated from a dot product $\boldsymbol{w}^T\boldsymbol{x}$, where $\boldsymbol{w} \in \mathbb{R}^k$ contains the weight associated with each feature;

3. A *learning algorithm* A that takes as input T and outputs L, using a learning process whose goal is to produce a model capable of accurately labeling instances from the same distribution as that of the training instances.

Through decades of research in the field of machine learning, a many different algorithms have been studied and tested for a wide variety of different NLP applications. Supervised learning algorithms for linear modeling can generally be divided into two groups: online learning algorithms and batch learning algorithms. Major online learning algorithms include the original perceptron (Rosenblatt, 1958) and large-margin extensions (Crammer and Singer, 2003; Crammer et al., 2006). Major batch learning algorithms include the maximum entropy approach (also referred to as logistic regression or log-linear modeling) (Berger et al., 1996) and support vector machines (SVM) (Cortes and Vapnik, 1995). In this thesis, our parsing algorithms use the batch SVM algorithm, which solves an optimization problem that regularizes $\boldsymbol{w}$ subject to the constraint that the hyperplane to which $\boldsymbol{w}$ is the normal separates positive and negative examples with sufficient margin.

An integral aspect of our learning algorithms is the use of a *kernel* approach, which allows for a non-linear feature space to be accessed while using only linear operations over the original feature space. We believe that this provides an elegant and computationally efficient way of avoiding an exhaustive hand-engineering of the feature space, which is often used to introduce non-linearity by including certain combinations of basic features. The *polynomial kernel* used in our experiments is especially useful due to its interpretability: given a degree d, it provides access to

non-linear space with dimensions corresponding to each of up to d-length combinations of features from the original space.

In Section 1.4.1 we formally explain the use of kernel functions in the supervised learning paradigm for linear models, and present the polynomial kernel function in particular. We then move on to an explanation of the learning algorithms used in this thesis, with Section 1.4.2 presenting the batch SVM algorithm for binary classification as well as extensions to multiclass and ranking settings. Finally, Section 1.4.3 discusses how categorical information typically found in NLP problems can be represented as real-valued features for linear models.

1.4.1 Kernels in Linear Models

A kernel function, as classically used in the context of machine learning for linear models, is a symmetric function $K(\boldsymbol{u}, \boldsymbol{v})$, where $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$, that satisfies Mercer's Theorem and defines a valid dot product in a feature space different from that of its input vectors (Cortes and Vapnik, 1995). By definition it therefore has a corresponding function ϕ such that

$$K(\boldsymbol{u}, \boldsymbol{v}) \equiv \phi(\boldsymbol{u})^T \phi(\boldsymbol{v}). \tag{1.2}$$

In many learning algorithms, and specifically in the perceptron and SVM algorithms, the norm $\boldsymbol{w}$ of the final separating hyperplane ends up being a linear combination of some number l of training vectors (so-called *support vectors*), which can be written as

$$\boldsymbol{w} = \sum_{i}^{l} y_i \alpha_i \boldsymbol{x}_i. \tag{1.3}$$

In order to perform learning and prediction in a non-linear space, we can apply the function ϕ throughout, which changes the final $\boldsymbol{w}$ to

$$\boldsymbol{w} = \sum_{i}^{l} y_i \alpha_i \phi(\boldsymbol{x}_i), \tag{1.4}$$

with a binary decision function that can be expressed using K as

$$sgn\Big(\boldsymbol{w}^T \phi(\boldsymbol{x})\Big) = sgn\left(\sum_{i=1}^{l} y_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x})\right). \tag{1.5}$$

Note that this non-linear decision function does not require any actual calculations in the non-linear space, allowing for $O(ln)$ time predictions (linear in n, the

size of the original space) for novel vectors as long as K can be computed efficiently. And as long as a learning algorithm can be implemented in such a way that the non-linear space does not need to be accessed either, this computational efficiency extends to the training process as well.

The Polynomial Kernel

Although there are many valid kernel functions that correspond to mappings onto potentially useful non-linear spaces, we focus exclusively on the *polynomial kernel* (Cortes and Vapnik, 1995) of the form

$$K(\boldsymbol{u}, \boldsymbol{v}) = (s\boldsymbol{u}^T\boldsymbol{v})^d, \tag{1.6}$$

where d is notably a parameter designating the degree of the polynomial. This kernel has the attractive property of operating over an easily understandable space of *combinations* of features from the original space. For example, if d=2 then we can view the non-linear feature space as having a dimension corresponding to each subset of size at most 2 of features u_i from the original space, as can be seen in the expansion[1] of ϕ:

$$\phi(\boldsymbol{u}) = \left\langle su_1^2, \sqrt{2}su_1u_2, \ldots, su_2^2, \sqrt{2}su_2u_3, \ldots, su_n^2 \right\rangle. \tag{1.7}$$

Earlier, we mentioned that kernel functions are useful only if they can be computed efficiently in the original space. The polynomial kernel satisfies this requirement by carrying out a dot product operation in the original space followed by a simple exponentiation.

1.4.2 Batch Learning with Kernel SVM

C-support vector classification (Boser et al., 1992; Cortes and Vapnik, 1995) is an approach to linear classification that combines three primary ideas:

1. A solution technique for obtaining an optimal large-margin separating hyperplane that expands the solution vector $\boldsymbol{w}$ onto support vectors;
2. The convolution of the dot-product using a kernel function, which extends the solution feature space from linear to non-linear;
3. The notion of soft margins, which allow for errors in the training set.

[1] This expansion is the only place in this thesis where a numeral vector subscript indicates a dimension rather than a position within a larger sequence of vectors.

The kernelized SVM for the binary case is posed as an optimization problem for the weight vector $\boldsymbol{w}$. The optimization problem simultaneously minimizes two terms: a term representing the extent to which $\boldsymbol{w}$ fails to correctly classify training instances beyond a certain margin, and a *regularization* term for the norm of $\boldsymbol{w}$. The problem description below closely follows Chang and Lin (2011):

$$\begin{aligned} \underset{\boldsymbol{w},\xi,b}{\text{minimize}} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i^l \xi_i \\ \text{subject to} \quad & y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0,\ i = 1, \ldots, l, \end{aligned} \tag{1.8}$$

where $\phi(\boldsymbol{x}_i)$ maps $\boldsymbol{x}_i$ into a higher-dimensional space, $C > 0$ is the regularization parameter, and b is the *bias*. Since the higher-dimensional space is typically much larger than the number of training examples l, the *dual* version of the optimization problem is solved,

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{minimize}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha} \\ \text{subject to} \quad & \boldsymbol{y}^T\boldsymbol{\alpha} = 0, \\ & 0 \leq \alpha_i \leq C,\ i = 1, \ldots, l, \end{aligned} \tag{1.9}$$

where $\boldsymbol{e} = [1, \ldots, 1]^T$ is a vector of all ones, Q is an l by l positive semidefinite matrix, $Q_{ij} \equiv y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and $K(\boldsymbol{x}_i, \boldsymbol{x}_j) \equiv \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$ is the kernel function. A solution can be obtained using a variety of techniques for solving dual quadratic problems, though a discussion of them is outside of the scope of this thesis.

Using the primal-dual relationship, the optimal weight vector $\boldsymbol{w}$ can be viewed as a linear combination of *support vectors*, or training vectors $\boldsymbol{x}_i$ with nonzero α_i,

$$\boldsymbol{w} = \sum_i^l y_i\alpha_i\phi(\boldsymbol{x}_i), \tag{1.10}$$

and the decision function can therefore be represented as

$$sgn(\boldsymbol{w}^T\phi(\boldsymbol{x}) + b) = sgn\left(\sum_{i=1}^l y_i\alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b\right). \tag{1.11}$$

As we noted before during our discussion of kernel functions, a crucial point is that the non-linear space never needs to be explicitly used, resulting in an effectively linear classifier that is nonetheless optimized over a non-linear space.

Multiclass SVM

In the *multiclass* setting for linear classification, the set of predicted labels is expanded to allow for more than two labels. Instead of $y_i \in \{-1, +1\}$, we now allow $y_i \in Y$, where Y is fixed-size set of labels.

A simple way to extend binary SVM classification to a multiclass setting is to use the *one-vs-one* approach:

1. For each pair of labels $y_i, y_j \in Y$ s.t. $i \neq j$, a binary SVM model $M_{i,j}$ is trained using the subset of training examples labeled with either y_i or y_j — examples with other labels are ignored.

2. To assign a label $\hat{y} \in Y$ to a novel vector $\boldsymbol{x}$, the algorithm uses a voting strategy that passes $\boldsymbol{x}$ through each binary model and selects the label with the most binary decisions in its favor.

Other methods of simulating the multiclass setting exist, such as the *one-vs-rest* approach where each $y_i \in Y$ is assigned a binary model trained with positive examples being those labeled as y_i and negative examples being those labeled as any $y_j \in Y - \{y_i\}$. We do not evaluate the *one-vs-rest* approach, though Hsu and Lin (2002) find that the two approaches are competitive.

Ranking SVM

In the *ranking* setting for linear classification, the ith ranking example consists of a sequence of m_i vectors $X_i = (\boldsymbol{x}_{i,1}, \ldots, \boldsymbol{x}_{i,m_i})$ and a (non-circular) set of ranking constraints R_i, with a constraint $r_{i,a,b}$ indicating that $\boldsymbol{x}_{i,a}$ should be ranked ahead of $\boldsymbol{x}_{i,b}$.

The goal is to ensure that ranking constraints for all examples are satisfied — that is, given the constraint $(\boldsymbol{x}_{i,a}, \boldsymbol{x}_{i,b})$, we want the weight vector $\boldsymbol{w}$ to satisfy

$$\begin{aligned} \boldsymbol{w}^T \phi(\boldsymbol{x}_{i,a}) - \boldsymbol{w}^T \phi(\boldsymbol{x}_{i,b}) &> 1 - \xi_{i,a,b} \\ \boldsymbol{w}^T (\phi(\boldsymbol{x}_{i,a}) - \phi(\boldsymbol{x}_{i,b})) &> 1 - \xi_{i,a,b} \end{aligned} \tag{1.12}$$

Noting the similarity to Equation 1.8, we can see that the ranking problem can essentially be posed as a binary problem, where each constraint $r_{i,a,b}$ corresponds to a binary example where the label is $+1$ and the vector in the non-linear feature space is $\phi(\boldsymbol{x}_{i,a}) - \phi(\boldsymbol{x}_{i,b})$. Joachims (2002) shows that the optimal weight vector can therefore be found in same way as described earlier for the binary case. We end up

with a final weight vector

$$\boldsymbol{w} = \sum_{i}^{l} \alpha_i \Big(\phi(\boldsymbol{x}_{i,a}) - \phi(\boldsymbol{x}_{i,b}) \Big), \tag{1.13}$$

and the decision function that ranks an input vector $\boldsymbol{x}$ with respect to another $\boldsymbol{x}'$ can be represented as

$$sgn\left(\sum_{i=1}^{l} \alpha_i \Big(K(\boldsymbol{x}_{i,a}, \boldsymbol{x}) - K(\boldsymbol{x}_{i,a}, \boldsymbol{x}') - K(\boldsymbol{x}_{i,b}, \boldsymbol{x}) + K(\boldsymbol{x}_{i,b}, \boldsymbol{x}') \Big) \right) \tag{1.14}$$

The subcase of ranking in which we are interested has a single correct vector in each example, with other vectors being deemed incorrect. If the ranking SVM model is being used to predict the *highest* ranking vector within a given set of candidates, one does not need to exhaustively calculate decisions between each pair of vectors, which would take quadratic time in the number of candidates. Instead, one can simply treat the ranking SVM model as a *scoring* function and select the vector $\boldsymbol{x}$ with the highest value

$$\underset{\boldsymbol{x}}{\text{argmax}}\ \boldsymbol{w}^T \boldsymbol{x} = \sum_{i=1}^{l} \alpha_i \Big(K(\boldsymbol{x}_{i,a}, \boldsymbol{x}) - K(\boldsymbol{x}_{i,b}, \boldsymbol{x}) \Big), \tag{1.15}$$

which takes only linear time. This property is useful for the multiple-candidate parsing and neighborhood correction algorithms we will present later on in Chapters 2 and 3, respectively, since these algorithms only need to find the highest scoring syntactic dependency from among a set of candidates.

1.4.3 Categorical Features

In NLP applications such as POS tagging and syntactic parsing, salient information for classification models often lies in the linguistic characteristics of a sentence, such as information describing its individual tokens. The features to which we have referred earlier in this section are real-valued, with each corresponding to a separate dimension in the feature space $\mathbb{R}^n$. In contrast, linguistic information tends to be *categorical*: a feature encoding the POS tag of a token would ideally take on categorical values among $\{\text{verb}, \text{noun}, \text{preposition}, \text{etc.}\}$. However, these values have no sensible analogues in the real space.

We follow a widely-used technique for representing categorical information in a real-valued feature space: each piece of categorical information has a correspond-

Categorical Information	Feature Template	Indicator Features
POS tag of current token in a sentence	t_{pos}	t_{pos}=verb, t_{pos}=noun, t_{pos}=preposition, ...
Lemma of current token in a sentence	t_{lem}	t_{lem}=être, t_{lem}=avoir, t_{lem}=aller, ...

Table 1.6: Examples of categorical information, feature template, and the indicator features that actually make up the feature space.

ing *feature template* that defines a set of indicator features in the real-valued space with each corresponding to a particular categorical value. Table 1.6 illustrates the relationship between a hypothetical piece of categorical information, its feature template, and the corresponding set of indicator features that would actually be used for model training and classification.

Chapter 2

Efficient Large-Context Dependency Parsing

Taken out of context I must seem so strange.

— Ani DiFranco

Having covered fundamental aspects of dependency syntax and machine learning in the previous chapter, this chapter formally presents the notion of dependency parsing and describes empirical results for two major families of parsers. Settling on the efficient transition-based parsing approach, we then begin the first main research thread of our thesis, which seeks to improve the amount of syntactic context in attachment decisions, by introducing a variant that simultaneously considers multiple governors when making attachments.

In Section 2.1 we formalize the problem of dependency parsing and investigate practical tradeoffs between algorithmic efficiency and the amount of contextual information available for dependency attachment decisions. To help along this investigation, we describe a preliminary study in which different dependency parsing approaches were tested for French, providing empirical evidence supporting our decision to focus exclusively on computationally efficient transition-based parsing.

In Section 2.2 we present the family of transition-based parsing algorithms that have been popularized in the past decade and which we use throughout the thesis. We discuss the existing ARC-STANDARD and ARC-EAGER algorithms within this family, and introduce ARC-EAGER-MC, a large-context variant that simultaneously considers multiple candidate governors for certain attachments during parsing.

Finally, Section 2.3 describes parsing experiments for French in which we compare the performance of ARC-STANDARD, ARC-EAGER, and ARC-EAGER-MC on the FTBDep, with a particular focus on the parsing accuracy of ambiguous syntactic constructions that are typically difficult to parse, namely prepositional phrase attachment and coordination.

2.1 Overview of Dependency Parsing

We are interested in computationally efficient approaches for *dependency parsing*, or the automatic prediction of syntactic dependency trees for sentences. This section is devoted to a formal presentation of the dependency parsing problem, as well as a brief introduction to different families of dependency parsing algorithms.

It should be noted that the most prominent work in statistical syntactic parsing, going as far back as the 1990's, has focused on phrase-structure parsing (Johnson, 1998; Collins, 2003; Klein and Manning, 2003). Widespread interest in dependency parsing began growing in the past decade, with algorithmic breakthroughs and seminal works for non-generative approaches (Nivre, 2003; Yamada and Matsumoto, 2003; McDonald et al., 2005) and prominent evaluation efforts such as the international CoNLL shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007a) that provided a platform for evaluating different approaches across multiple languages.

We organize our discussion as follows: Section 2.1.1 provides a formal description of the dependency parsing problem, in a way that mirrors the smaller scale machine learning problem for linear models, and reiterates the graph requirements for tree structures according to dependency grammar using formal mathematical notation. Section 2.1.2 then briefly summarizes an initial benchmarking experiment for French dependency parsing in which we participated, connecting our findings with those at large for dependency parsing approaches across different languages. This allows us to introduce the major families of dependency parsing algorithms and discuss their relative advantages and disadvantages, leading us to a decision on which to use as the base approach for our thesis.

2.1.1 Formalizing Data-Driven Dependency Parsing

A parser is a system that is capable of analyzing the syntax of a sentence, outputting a tree corresponding to the sentence that best adheres to the grammar of its language. Though hand-built symbolic parsers have a longer history of use, notably for French (Balfourier et al., 2005), current research in NLP focuses almost exclusively on data-driven statistical parsing.

In comparing a data-driven statistical approach to symbolic one, we can identify a number of advantages and disadvantages. A key advantage of the data-driven statistical approach is its robustness for sentences that do not strictly conform to the grammar of a language. This might occur in spoken dialogue or in non-canonical text produced in contemporary web culture, the latter of which is a growing area of study in NLP (Petrov and McDonald, 2012). Another key advantage is that a statistical approach is particular well-suited to handling the problem of syntactic ambiguity, which we have previously discussed (cf. Section 1.2.3). Through the introduction of probabilities for derivation rules or of machine learning for classification, statistical parsers inherently disambiguate between possible parse trees for a sentence. One major downside of data-driven statistical approaches is their reliance on a substantial amount of training data needed to learn accurate parsing models, with this data typically requiring manual syntactic annotation; symbolic approaches notably do not have this requirement. On the other hand, symbolic approaches are labor intensive in their own way by requiring the difficult work of defining the syntactic rules and constraints covering the entire grammar for a language.

In this thesis, we have ultimately chosen to focus on a data-driven, statistical machine learning approach to dependency parsing. The dependency parsing problem can thus be viewed as including four elements, adapted from the definitions of Kübler et al. (2009), with the use of *arc* instead of edge to match their terminology:

1. A *functional role label set* $R = \{r_1, \ldots, r_m\}$, which is defined as a finite set of possible labels for dependency arcs between pairs of words.

2. A *training set* D consisting of a sequence of tuples of the form (x_i, G_i). Within D, a *sentence* x is defined as a sequence of word tokens $w_0 w_1 \ldots w_n$ with w_0 serving as a dummy ROOT token. A *dependency graph* $G = (V_x, A)$ for x is a labeled directed graph with nodes $V_x = \{w_0, w_1, \ldots, w_n\}$ and arcs $A \subseteq V_x \times R \times V_x$. For each training tuple (x, G), G must be a dependency tree with respect to x and to R.

3. A *parsing model* M that predicts a well-formed dependency tree G for an input sentence x.

4. A *learning algorithm* that takes as input D and outputs M, using a learning process whose goal is to produce a model capable of accurately producing syntactic dependency analyses of sentences from the same distribution as those of the training set.

A key requirement of dependency training sets and parsing models is that their dependency graphs must be well-formed dependency trees adhering to the requirements which we have previously outlined in more general terms in Section 1.2.2. We will now reiterate these dependency structure requirements using notation consistent with the description of dependency parsing above. For simplicity, we introduce the following additional shorthand notation, as in Kübler et al. (2009):

- Dependency relation: $w_i \rightarrow w_j \equiv (w_i, r, w_j) \in A$ for some $r \in R$, with w_i being the governor of w_j.
- Reflexive transitive closure of a dependency relation: $w_i \rightarrow^* w_j \equiv (i = j) \vee (w_i \rightarrow^* w_{i'} \wedge w_{i'} \rightarrow w_j)$ for some $w_{i'} \in V$.
- Undirected dependency relation: $w_i \leftrightarrow w_j \equiv w_i \rightarrow w_j \vee w_j \rightarrow w_i$.
- Reflexive transitive closure of an undirected dependency relation: $w_i \leftrightarrow^* w_j \equiv (i = j) \vee (w_i \leftrightarrow^* w_{i'} \wedge w_{i'} \leftrightarrow w_j)$ for some $w_{i'} \in V$.

Apart from the binary, directed, and labeled requirements for dependencies that are already explicit in our definition of dependency graphs above, the following tree requirements are also imposed (cf. Section 1.2.2) because the dependency graphs we consider are projective trees with root w_0:

- *anti-reflexivity*: There does not exist $w_i \in V$ such that $w_i \rightarrow w_i$.

- *connectedness*: For all $w_i, w_j \in V$ it is the case that $w_i \leftrightarrow^* w_j$.
- *acyclicity*: For all $w_i, w_j \in V$, if $w_i \rightarrow w_j$ then it is not the case that $w_j \rightarrow^* w_i$.
- *unique governance*: For all $w_i, w_j \in V$, if $w_i \rightarrow w_j$ then there does not exist $w_{i'} \in V$ such that $i' \neq i \wedge w_{i'} \rightarrow w_j$.
- *root governance*: There does not exist $w_i \in V$ such that $w_i \rightarrow w_0$.
- *projectivity*: Every arc $(w_i, r, w_j) \in A$ is projective. Formally, $w_i \rightarrow^* w_k$ for all $i < k < j$ if $i < j$, or for all $j < k < i$ if $j < i$.

2.1.2 Lessons from a French Parsing Benchmark

We now turn to the discussion of a benchmarking experiment for French parsing in which we participated (Candito et al., 2010b), and whose results had a major influence on the strategy adopted in this thesis with respect to the choice of parser and features. The benchmarking experiment was intended as an initial, broad evaluation of different statistical dependency parsing approaches for French, following the automatic conversion of the FTBDep from the FTB carried out by Candito et al. (2010a). The goal of the benchmarking experiment was to compare the parsing accuracy and computational efficiency of different parsing approaches for French, in the same vein as previous focused works for English (Cer et al., 2010) and for German (Kübler, 2008). While an important first step in our research, the trends and lessons found in this benchmarking experiment for French mainly confirm what had already been found in the seminal CoNLL shared task across 13 different languages (Buchholz and Marsi, 2006), though that evaluation notably did not include French.

Although the benchmarking experiment included a popular phrase-structure parsing approach, with post-processing to convert output trees into dependency structure for comparative evaluation, we choose not to include that approach in our following discussion. Instead, we focus exclusively on the two major families of data-driven dependency parsers using very distinct approaches that have been favored in the NLP literature for different languages over the last decade, and which were tested in the benchmark. Below we describe these two parser families, followed by a presentation of the evaluation and results that motivated the choice of dependency parsing algorithm for this thesis.

Dependency Parser Families

The first major family of data-driven dependency parsing, *transition-based parsing*, reduces the problem of parsing a sentence to the problem of finding an optimal

sequence of transitions through an abstract transition system, traditionally maintaining a partially-built parse tree as well as a stack, buffer, or other structures used by the algorithm. With seminal works by Yamada and Matsumoto (2003) and by Nivre (2003), transition-based parsers are typically trained in a *locally-optimized* manner: the output transition sequence, which defines the parse tree for a sentence, is obtained using an *oracle* function implemented as a classifier that predicts each subsequent transition using *history-based* features that reflect past transitions. While no guarantees are made as to the overall optimality of the final transition sequence, the fact that decisions are made locally allows for $O(n)$ parsing time complexity with respect to the length of the sentence. Note that we will provide a more detailed description of the transitioned-based parsing approach later in this chapter.

The second approach is termed *graph-based parsing*, and it works by operating directly on graph structures. It initializes a dependency graph with nodes corresponding to word tokens from a sentence, and then predicts a *globally-optimized* set of arcs that constitute the output dependency parse tree. Each possible dependency graph for a sentence can be assigned a score that corresponds to the sum of the scores of its *factors*, which consist of one or more linked arcs. While large factors are desirable in order to capture sophisticated linguistic constraints, they come at the expense of increased time complexity; however, it should be noted that even when using the smallest size factors these algorithms have a large $O(n^3)$ parsing time complexity with respect to the length of the sentence. More precisely, adaptations of Eisner's algorithm (Eisner, 1996) for finding the optimal parse have $O(n^3)$ complexity when using 1-arc factors (McDonald et al., 2005) or sibling 2-arc factors (McDonald and Pereira, 2006), and complexity increases to $O(n^4)$ when using generic 2-arc factors (Carreras, 2007) or 3-arc factors (Koo and Collins, 2010).

Evaluation Setup

Evaluation of the different parsing approaches was performed using a training, development, and test set split of the FTBDep, as well as preprocessing of the FTBDep with automatic POS tagging using the MElt package (Denis and Sagot, 2009) and lemmatization using the Lefff lexicon (Sagot, 2010). Because we describe the same preprocessing setup for experiments later in this chapter, we refrain from going further into detail in this section.

In order to compare the two dependency parsing approaches fairly, Candito et al. (2010b) used state-of-the-art parsing systems and parameters that had already achieved results for other languages competitive with the best known reported results. For transition-based parsing, the freely-available MaltParser system (Nivre et al., 2007b) was used. The transition system used was ARC-EAGER, which we will

Setting	FTBDep dev		FTBDep test	
	LAS	**UAS**	**LAS**	**UAS**
MaltParser	86.2	89.0	86.7	89.3
MSTParser	87.2	90.0	87.6	90.3

Table 2.1: Labeled (LAS) and unlabeled (UAS) attachment scores of parsers on the FTBDep development and test sets.

Setting	Running time
MaltParser	00:58
MSTParser	14:12

Table 2.2: Running times (min:sec) of parsers on the FTBDep development set on an iMac 2.66 GHz computer.

describe in more detail later in this chapter, and the multiclass classifier for learning locally-optimal transitions was trained using the LIBLINEAR package (Fan et al., 2008). The set of feature templates was tuned using the FTBDep development set. For graph-based parsing, the freely-available MSTParser system (McDonald et al., 2005) was used. The settings included sibling 2-arc factors, with projective decoding to ensure projective output trees. Additionally, functional role labeling was carried out as a post-processing sequence classification step, following McDonald et al. (2006). Feature templates were those from the software's default settings.

Results

Results were obtained after the two parsers were trained on the FTBDep training set, then used to automatically parse the FTBDep development and test sets. Table 2.1 reproduces the main results of the evaluation, as reported by Candito et al. (2010b). The evaluation metric is the proportion of word tokens that are assigned a dependency arc with the correct governor; *labeled attachment score* (LAS) requires that the arc have the correct functional role label for it to be considered correct, while *unlabeled attachment score* (UAS) ignores whether the label is accurate or not. We can see that MSTParser obtains the higher parsing results, both LAS and UAS, compared to MaltParser. While the differences in performance are statistically significant, the scores differ by a modest 1 point each of LAS and UAS.

In order to roughly gauge the effects of different time complexities for different parsers, each parser's running time was also recorded. Table 2.2 reproduces the running time results. In this part of the evaluation, we find that MaltParser, which has an $O(n)$ theoretical parsing time, predictably runs at over an order of magnitude faster than MSTParser, which has a theoretical $O(n^3)$ parsing time.

Another aspect of the evaluation was to test the use of lemmas and morphological features, with two relevant modifications: the replacement of each inflected word form with its automatically predicted lemma, and the introduction of morphological features for MaltParser and MSTParser. Both of these modifications led to increased LAS and UAS scores, though the improvements were small for both MaltParser and MSTParser (ranging from 0.1 to 0.4 points of LAS and UAS).

Choice of Dependency Parsing Family

The benchmarking results were fairly decisive in terms of showing the tradeoffs between using the two major approaches to dependency parsing. Taking a wider view, these results are also mostly the same across different languages, as demonstrated in the CoNLL-X shared task on multilingual dependency parsing (Buchholz and Marsi, 2006) across 13 different languages (with French not included among them). In the results for that shared task, the two implementations that performed best across the board were those by the leading researchers for graph-based parsing (McDonald et al., 2006) and for transition-based parsing (Nivre et al., 2006); graph-based parsing was more accurate overall, but the difference was small. Given the agreement of wider results in the literature across different languages with those we obtained for French, we had solid evidence with which to decide which parsing approach to use as the basis for the work of our thesis.

Our decision was informed mainly by the tradeoff between parsing accuracy and computational efficiency for different parsing families. As indicated above, the efficient linear-time transition-based parser had a running time an order of magnitude faster than that of the rival cubic-time graph-based parser. While the more complex approach achieved slightly higher parsing accuracies, we felt that the difference was not large enough to outweigh the substantial discrepancy in computational efficiency. We believe that, given the rate of technological progress and the ever increasing amount of natural language data that is available for analysis and processing, high speed will become a necessity for wide-use syntactic parsers in NLP applications. We thus decided to focus this thesis on methods for improving transition-based parsing.

Ideas for Improving Transition-Based Parsing

A natural place to begin the search for ways in which transition-based parsing could be improved was to examine the reasons behind the differences in accuracy of the transition-based parsing approach compared to the more complex graph-based parsing approach. This issue has been previously investigated in the NLP literature,

most relevantly in the work of McDonald and Nivre (2007) that compares errors made by the two major dependency parsing approaches in the CoNLL-X shared task (Buchholz and Marsi, 2006). We thus lean on their findings as well as our intuition on how they might extend to French parsing, though we did not conduct a corresponding extensive error analysis for our benchmarking results.

Due to its derivation of parse trees in a deterministic manner using limited local context, transition-based parsing can be found lacking in its treatment of ambiguity at a sentential or even simply a non-local level. While graph-based parsing does use information at the local level, as its scores are necessarily restricted to limited-size edge factors, it is crucially able to synthesize local information into a global metric that can account for low-scoring local areas that nonetheless reside in the optimal tree. This is likely a reason for the finding of McDonald and Nivre (2007) that long-distance dependencies were more accurately parsed by the graph-based parser than by the transition-based parser.

The first major goal of our work is thus to increase the context available during local decisions for transition-based parsing, while trying at the same time to retain computational efficiency. We noted earlier that prepositional phrase attachment and coordination were difficult phenomena to parse accurately; these can be seen as potentially non-local linguistic phenomena with high levels of syntactic ambiguity, which means that their attachment decisions could presumably benefit from more context during parsing. Efficient methods for introducing large-context into attachment decisions are explored later in this chapter as well as in Chapter 3.

Another take-away point from the benchmarking evaluation was the idea that work at the lexical level still had potential for improving parsing results. The replacement of word forms with lemmas and the use of morphological features improved parsing accuracy at little computational expense. We decided that the replacement of word forms with other generalized lexical classes, as well as the inclusion of features encoding lexical subcategorization and preference, would potentially provide an additional way to improve the handling of ambiguity in transition-based parsing while retaining computational efficiency. Chapters 4 and 5 are devoted to the exploration of these lexical approaches.

2.2 Transition-Based Parsing

Having decided to work within the family of transition-based parsers following our preliminary benchmarking experiment for French, in this section we now describe the formal characteristics of the transition-based approach to parsing, including descriptions of the specific algorithms used and their theoretical guarantees.

As previously mentioned, transition-based parsing reduces the problem of parsing a sentence to the problem of finding an optimal sequence through an abstract transition system, traditionally maintaining a partially-built parse tree as well as secondary stack and buffer structures used by the algorithm. It should also be noted that transition-based dependency parsers can be thought of as successors to previous related work on shift-reduce phrase-structure parsing (Aho et al., 1986), though we will not discuss those approaches here. In our following discussion, we primarily follow the research of Nivre and colleagues (Nivre, 2003; Nivre et al., 2006; Nivre, 2008; Kübler et al., 2009), with terminology and notation closely following those works.

2.2.1 The Generalized Framework

We present the transition-based approach to dependency parsing as an elaboration of the elements described in Section 2.1.1 for generalized dependency parsing. The basic functional role label set and training set elements remain the same, while the parsing model and learning algorithm are implemented in a manner specific to a transition-based approach. As we briefly mentioned in our discussion of the benchmarking experiment in Section 2.1.2, the parsing model is a transition system that maintains a partially-built graph structure as well as a stack and buffer while navigating from initial to terminal configurations.

Formally, following the definition of Nivre (2008), a *transition system* is a quadruple $S = (C, T, f_I, C_t)$, where:

1. C is a set of *configurations*, each of which contains a set A of dependency arcs and a fixed set intermediary structures depending on the algorithm variant. All variants that we are aware of have at least a buffer β of remaining word forms;

2. T is a small set of transitions, each of which is a (partial) function $t : C \rightarrow C$;

3. f_I is an initialization function, which maps a sentence $x = w_0 w_1 \ldots w_n$ to a starting configuration c_0 where $\beta_{c_0} = [1, \ldots, n]$;

4. $C_t \subseteq C$ is a set of terminal configurations. For the transition systems that we consider, C_t is the set of all configurations c_i where the buffer is empty, or $\beta_{c_i} = [\,]$.

Additionally, a *transition sequence* in S for a sentence $x = w_0 w_1 \ldots w_n$ is a sequence $C_{0,m} = c_0 c_1 \ldots c_m$ of configurations, such that:

1. $c_0 = f_I(x)$;
2. $c_m \in C_t$;
3. for every i from $1 \leq i \leq m$, $c_i = t(c_{i-1})$ for some $t \in T$.

The parsing algorithm for a transition-based parser takes an input sentence x and applies a sequence of transitions from the initial configuration $f_I(x)$ in such a way that the terminal configuration c_m corresponds to a predicted dependency tree $G_{c_m} = (V_x, A_{c_m})$. In order to determine which sequence of transitions to follow, the parsing algorithm uses an *oracle* function $o : C \rightarrow T$ that determines which transition to take given a particular configuration.

The learning algorithm in transition-based parsing is tasked with optimizing the oracle: we want an o such that, for each pair (x, G) in the training set D, it will best guide x through S from the initial configuration $f_I(x)$ to a *gold* terminal configuration $c_g \in C_t$, where c_g is defined as gold if and only if $G_{c_g} = G$. In order to learn an appropriate oracle, a transition classification training set D' must be derived from the original sentential training set D. Each tuple $(x, G) \in D$ is converted into a sequence of training tuples (c_i, t_i) that corresponds to a correct transition sequence, which we define as a transition sequence $C_{0,g}$ leading from $f_I(x)$ to the correct c_g; there are potentially multiple correct transition sequences for any given training tuple, so as we will discuss later a consistent method should be defined to derive a unique *gold transition sequence* for any given training tuple. A multiclass classification learner can then be viewed as training o to accurately predict, given history-based features encoding information about the current configuration, the correct transition to take. Note that this learning setup leads to an oracle that is locally optimized, as the oracle predicts individual transitions rather than joint sequences of them.

Incrementality and Correctness

There are two additional properties identified by Nivre (2008) as key for transition-based parsers. The first is *incrementality*, which guarantees that a transition-based parser will terminate. Formally, a transition system $S = (C, T, f_I, C_t)$ is incremental if and only if, for every configuration $c \in C$ and transition $t \in T$, it holds that:

1. The buffer must never grow in size: $|\beta_c| \geq |\beta_{t(c)}|$.
2. Parsing must terminate when the buffer is empty: if $\beta_c = [\,]$, then $c \in C_t$.
3. Arcs can never be removed: if $a \in A_c$, then $a \in A_{t(c)}$.

The second property is that of *correctness* with respect to a particular class $\mathbb{G}$ of dependency graphs. This property requires that a transition system for parsing be both *sound*, meaning it only derives parses in $\mathbb{G}$, and *complete*, meaning that it is capable of deriving all parses in $\mathbb{G}$. Formally, given a transition system $S = (C, T, f_I, C_t)$ and a class $\mathbb{G}$ of dependency graphs:

1. S is sound for $\mathbb{G}$ if and only if, for every sentence x and every transition sequence $C_{0,m}$ for x in S, the parse $G_{c_m} \in \mathbb{G}$.

2. S is complete for $\mathbb{G}$ if and only if, for every sentence x and every dependency graph G_x for x in $\mathbb{G}$, there is a transition sequence $C_{0,m}$ for x in S such that $G_{c_m} = G_x$.

3. S is correct for $\mathbb{G}$ if and only if it is sound and complete for $\mathbb{G}$.

Major Transition-Based Parsers

A variety of parsers that can be characterized using the above generalized transition-based parsing framework have been explored in the past. The traditional approaches of Nivre (2003) and of Yamada and Matsumoto (2003) use, in addition to a buffer of untreated word forms, a stack to store partially processed word forms. The transition sets for these approaches correspond roughly to classic notions of 'shift' and 'reduce', and have the restriction of deriving projective dependency trees. There have also been list-based approaches to transition-based parsing, notably those of Covington (2001) and of Nivre (2007), which use open lists for partially processed word forms and are capable of deriving non-projective dependency trees.

We focus our attention on the traditional approaches, due to the fact that the FTBDep contains only projective trees, so there is less need to allow for non-projectivity. The two specific approaches we consider are the ARC-EAGER and ARC-STANDARD parsers of Nivre (2008). In Section 2.2.2, we describe these two algorithms along with their corresponding theoretical guarantees regarding soundness and completeness. In Section 2.2.3, we then present a variant of ARC-EAGER that we term ARC-EAGER-MC, whose purpose is to introduce additional context and treat the problem of parse ambiguity by simultaneously considering multiple candidate governors for the right-arc transition. We also present theoretical results for the soundness and completeness of the ARC-EAGER-MC algorithm, analogous to those of Nivre (2008) for ARC-EAGER and ARC-STANDARD.

2.2.2 Existing Approaches: ARC-STANDARD, ARC-EAGER

We now present the ARC-STANDARD and ARC-EAGER approaches, the two major implementations of the transition-based dependency parsing framework that we have chosen to use as the basis for our experiments for French, and for which we will also explore a larger-context variant.

In both of these approaches, the ancillary data structures for parsing are limited to the previously mentioned buffer β of remaining word forms and additionally a stack σ of processed word forms. We can thus make more precise the definition of a configuration for these two approaches: each configuration is of the form $c = (A, \beta, \sigma)$, where A is a set of arcs, β is a buffer, and σ is a stack. In our formulations, we choose not to use the dummy w_0 directly during parsing; instead, we apply a final action after reaching a terminal state in which all word forms that have not been assigned governors are set as dependents of w_0 with the ROOT label. The initialization function f_I for a sentence $x = w_0 w_1 \dots w_n$ then produces the configuration where $A = \{\}$, $\beta = [1, \dots, n]$, and $\sigma = []$. Note that for ease of representation, both the stack and buffer are shown as lists with the top of the stack and the first item in the buffer appearing in leftmost position. Finally, we employ simple shorthand operations over structures $add(\cdot, w_i)$, which represents shifting onto the front of a buffer or pushing onto a stack, and $rem(\cdot)$, which represents deleting from the front of a buffer or popping from a stack.

We will discuss below the choice of transition set, which is the sole detail on which the ARC-STANDARD and ARC-EAGER approaches differ, and also briefly describe their theoretical guarantees, as analyzed by Nivre (2008). Subsequently, we will discuss the possible relative advantages of each approach for effectively modeling linguistic phenomena in French dependency syntax.

The ARC-STANDARD Transition Set

In the ARC-STANDARD implementation of transition-based dependency parsing framework, the set of transitions used by the transition model contains three types of transitions that move from a previous configuration $c_i = (A_i, \beta_i, \sigma_i)$ to a new configuration $c_j = (A_j, \beta_j, \sigma_j)$:

1. LEFT-ARC(l) (for a functional role label l) adds a new arc from the word form at the front of the buffer to the word form at the top of the stack, then pops the stack. Formally, $A_j = A_i \cup \{(\beta_i[0], l, \sigma_i[0])\}$, $\beta_j = \beta_i$, and $\sigma_j = rem(\sigma_i)$. As a precondition for taking this type of transition, the word form on top of the stack must not be the artificial root note and must not already have a governor. Formally, $\sigma_i[0] \neq w_0$ and $(\cdot, \cdot, \sigma_i[0]) \notin A_i$.

2. RIGHT-ARC(l) (for a functional role label l) adds a new arc from the word form at the top of the stack to the word form at the front of the buffer, then replaces the word from at the front of the buffer with the word form at the top of the stack.[1] Formally, $A_j = A_i \cup \{(\sigma_i[0], l, \beta_i[0])\}$, $\beta_j = add(rem(\beta_i), \sigma_i[0])$, and $\sigma_j = rem(\sigma_i)$. As a precondition for taking this type of transition, the word form at the front of the buffer must not already have a governor. Formally, $(\cdot, \cdot, \beta_i[0]) \notin A_i$.

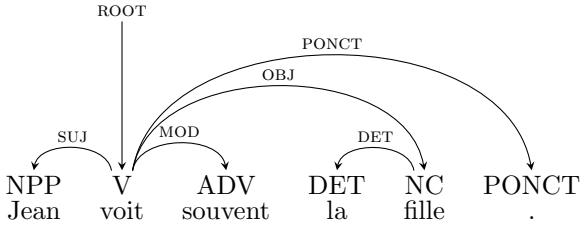
3. SHIFT moves the word form at the front of the buffer onto the top of the stack. Formally, $A_j = A_i$, $\beta_j = rem(\beta_i)$, and $\sigma_j = add(\sigma_i, \beta_i[0])$. There are no preconditions for taking this type of transition.

For the derivation of a gold transition sequence for a sentence, which is required for training the transition system oracle, transitions are selected at each configuration based on an order of descending priority, with arc-building transitions applied whenever possible (note that both cannot be simultaneously possible) and SHIFT consequently having the lowest priority. Aside from the above basic preconditions for each transition, the LEFT-ARC(l) transition requires that the created arc is part of the gold dependency tree for the sentence, and the RIGHT-ARC(l) transition further requires that the dependent have all of its own outgoing gold arcs already present in the configuration. Figure 2.1 illustrates the ARC-STANDARD gold transition sequence for the sentence "Jean voit souvent la fille" ("Jean sees often the girl"), with the gold dependency tree for the sentence shown on top.

For the theoretical guarantees of ARC-STANDARD, Nivre (2008) shows that ARC-STANDARD fulfills the requirements of being correct with respect to the class of projective dependency trees, and that it's worst-case computational complexity is $O(n)$ in the length of the input sentence for both time and space.

As Nivre (2008) notes in his discussion, ARC-STANDARD uses a very simple and traditional approach that is closely related to shift-reduce parsers for context-free grammars (Aho et al., 1986). The SHIFT action serves the same purpose in both approaches, while the arc building actions are equivalent to the reductions made in CFG parsing, replacing a head-dependent structure with its head. As such, ARC-STANDARD is perhaps the most natural approach for transition-based dependency parsing.

[1] Kübler et al. (2009) note that the RIGHT-ARC(l) transition may seem counterintuitive, given that the buffer is meant to contain word forms that have not yet been processed. However it is necessary to put this word form back on the buffer from the stack in order to allow it to attach to a governor on its left.



Transitions	Configurations
INITIALIZE	β = [‘Jean’, ‘voit’, ‘souvent’, ‘la’, ‘fille’, ‘.’], σ = [], A = {}
SHIFT	β = [‘voit’, ‘souvent’, ‘la’, ‘fille’, ‘.’], σ = [‘Jean’], A = {}
LEFT-ARC(SUJ)	β = [‘voit’, ‘souvent’, ‘la’, ‘fille’, ‘.’], σ = [], A = {(‘voit’, SUJ, ‘Jean’)}
SHIFT	β = [‘souvent’, ‘la’, ‘fille’, ‘.’], σ = [‘voit’], A = {(‘voit’, SUJ, ‘Jean’)}
RIGHT-ARC(MOD)	β = [‘voit’, ‘la’, ‘fille’, ‘.’], σ = [], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’)}
SHIFT	β = [‘la’, ‘fille’, ‘.’], σ = [‘voit’], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’)}
SHIFT	β = [‘fille’, ‘.’], σ = [‘la’, ‘voit’], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’)}
LEFT-ARC(DET)	β = [‘fille’, ‘.’], σ = [‘voit’], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’), (‘fille’, DET, ‘la’)}
RIGHT-ARC(OBJ)	β = [‘voit’, ‘.’], σ = [], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’), (‘fille’, DET, ‘la’), (‘voit’, OBJ, ‘fille’)}
SHIFT	β = [‘.’], σ = [‘voit’], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’), (‘fille’, DET, ‘la’), (‘voit’, OBJ, ‘fille’)}
RIGHT-ARC(PONCT)	β = [‘voit’], σ = [], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’), (‘fille’, DET, ‘la’), (‘voit’, OBJ, ‘fille’), (‘voit’, PONCT, ‘.’)}
SHIFT	β = [], σ = [‘voit’], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’), (‘fille’, DET, ‘la’), (‘voit’, OBJ, ‘fille’), (‘voit’, PONCT, ‘.’)}
TERMINATE	A_{final} = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’), (‘fille’, DET, ‘la’), (‘voit’, OBJ, ‘fille’), (‘voit’, PONCT, ‘.’), (ROOT, ROOT, ‘voit’)}

Figure 2.1: Gold ARC-STANDARD transition sequence, with corresponding intermediate configurations, for the sentence: “Jean voit souvent la fille.” (“Jean sees often the girl.”)

The ARC-EAGER Transition Set

In the ARC-EAGER implementation of the transition-based dependency parsing framework, the set of transitions used by the transition model contains four types of transitions that move from a previous configuration $c_i = (A_i, \beta_i, \sigma_i)$ to a new configuration $c_j = (A_j, \beta_j, \sigma_j)$. We note that the LEFT-ARC(l) and SHIFT transitions are the same as before for ARC-EAGER, while the RIGHT-ARC(l) transition has a slightly different behavior and a new REDUCE transition is introduced. The following are the set of transitions for ARC-EAGER:

1. LEFT-ARC(l) (for a functional role label l) adds a new arc from the word form

at the front of the buffer to the word form at the top of the stack, then pops the stack. Formally, $A_j = A_i \cup \{(\beta_i[0], l, \sigma_i[0])\}$, $\beta_j = \beta_i$, and $\sigma_j = \mathit{rem}(\sigma_i)$. As a precondition for taking this type of transition, the word form on top of the stack must not be the artificial root note and must not already have a governor. Formally, $\sigma_i[0] \neq w_0$ and $(\cdot, \cdot, \sigma_i[0]) \notin A_i$.

2. RIGHT-ARC(l) (for a functional role label l) adds a new arc from the word form at the top of the stack to the word form at the front of the buffer, then moves the word form at the front of the buffer onto the top of the stack. Formally, $A_j = A_i \cup \{(\sigma_i[0], l, \beta_i[0])\}$, $\beta_j = \mathit{rem}(\beta_i)$, and $\sigma_j = \mathit{add}(\mathit{rem}(\sigma_i), \beta_i[0])$. As a precondition for taking this type of transition, the word form at the front of the buffer must not already have a governor. Formally, $(\cdot, \cdot, \beta_i[0]) \notin A_i$.

3. REDUCE simply pops the stack. Formally, $A_j = A_i$, $\beta_j = \beta_i$, and $\sigma_j = \mathit{rem}(\sigma_i)$. As a precondition for taking this type of transition, the word form on top of the stack must have a governor. Formally, $(\cdot, \cdot, \sigma_i[0]) \in A_i$.

4. SHIFT moves the word form at the front of the buffer onto the top of the stack. Formally, $A_j = A_i$, $\beta_j = \mathit{rem}(\beta_i)$, and $\sigma_j = \mathit{add}(\sigma_i, \beta_i[0])$. There are no preconditions for taking this type of transition.

For the derivation of a gold transition sequence for a sentence, transitions are selected at each configuration based on an order of descending priority, similar to that of ARC-STANDARD, with arc-building transitions having higher priority. This time, both LEFT-ARC(l) and RIGHT-ARC(l) simply require that the created arc is part of the gold dependency tree for the sentence. Also, REDUCE requires that the configuration contains all arcs in the gold dependency tree for which the word form at the top of the stack is governor. It is important to note here that during gold oracle parsing, there exist two different ways of prioritizing SHIFT compared to REDUCE that correspond to adequate oracles for deriving gold trees, with the result being slightly different transition sequences. We have not found this detail in previous discussions in the literature concerning ARC-EAGER. The two prioritization orders are as follows:

1. Do REDUCE if possible. Else do SHIFT.

2. Do SHIFT if the word form at the front of the buffer does not have its gold governor to its left (in which case it would be left stranded on the stack). Else do REDUCE if possible. Else do SHIFT.

While the first ordering appears to be simpler, we have found that the second ordering is the one used in MaltParser, the official software implementation of ARC-EAGER transition-based parsing (Nivre et al., 2007b). Theoretically, it is not clear to us whether one of the orderings produces transition sequences that are more suitable than the other for learning or parsing. It can be noted that the second ordering uses REDUCE only when no other option is available, which means that possible governors on the stack (for dependents on the buffer) will remain accessible longer during parsing. Ultimately, we choose to use the second ordering for two reasons: the fact that it appears to be more accepted, and more importantly the fact that its lazy use of REDUCE makes it easier to compare theoretically to the variant we will be presenting shortly. Figure 2.2 illustrates the ARC-EAGER gold transition sequence using the second ordering for the sentence "Jean voit souvent la fille" ("Jean sees often the girl"), with the gold dependency tree for the sentence shown on top.

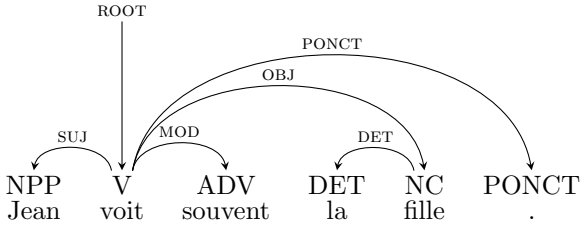
For the theoretical guarantees of ARC-EAGER, Nivre (2008) shows that ARC-EAGER, like ARC-STANDARD, fulfills the requirements of being correct with respect to the class of projective dependency trees, and it's worst-case computational complexity is also $O(n)$ in the length of the input sentence for both time and space.

As Nivre (2008) notes in his discussion, ARC-EAGER adds arcs for right dependents as soon as possible; that is, before the right dependent has found all of its own dependents, as was the case for ARC-STANDARD. As a consequence, the RIGHT-ARC(l) transition cannot replace a head-dependent structure with its head, but must store both the head and the dependent on the stack for further processing. This necessitates the addition of an explicit REDUCE transition, which allows the transition system to complete the reduction of the head-dependent structure.

Comparing the two approaches

We now present a brief comparison of the ARC-STANDARD and ARC-EAGER approaches. The key difference is the special constraint imposed on ARC-STANDARD for RIGHT-ARC(l): the dependent needs to have all of its own dependents attached before this transition can be taken. As noted above, the fact that ARC-EAGER eschews this constraint leads to the addition of a REDUCE transition. Given these differences, what advantages and drawbacks might result from using ARC-EAGER instead of ARC-STANDARD?

On one hand, ARC-EAGER has the drawback compared to ARC-STANDARD of complicating oracle decisions by adding a fourth transition. If an oracle classifier is required to choose between four transitions instead of three, it could potentially make learning an accurate oracle more difficult. Another possible drawback of ARC-

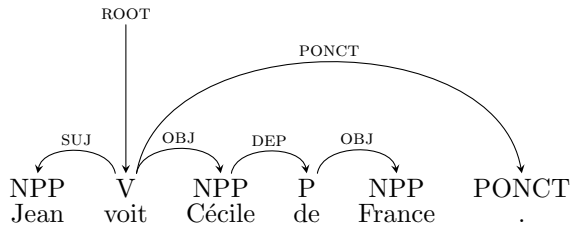


Transitions	Configurations
INITIALIZE	β = [‘Jean’, ‘voit’, ‘souvent’, ‘la’, ‘fille’, ‘.’], σ = [], A = {}
SHIFT	β = [‘voit’, ‘souvent’, ‘la’, ‘fille’, ‘.’], σ = [‘Jean’], A = {}
LEFT-ARC(SUJ)	β = [‘voit’, ‘souvent’, ‘la’, ‘fille’, ‘.’], σ = [], A = {(‘voit’, SUJ, ‘Jean’)}
SHIFT	β = [‘souvent’, ‘la’, ‘fille’, ‘.’], σ = [‘voit’], A = {(‘voit’, SUJ, ‘Jean’)}
RIGHT-ARC(MOD)	β = [‘la’, ‘fille’, ‘.’], σ = [‘souvent’, ‘voit’], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’)}
SHIFT	β = [‘fille’, ‘.’], σ = [‘la’, ‘souvent’, ‘voit’], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’)}
LEFT-ARC(DET)	β = [‘fille’, ‘.’], σ = [‘souvent’, ‘voit’], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’), (‘fille’, DET, ‘la’)}
REDUCE	β = [‘fille’, ‘.’], σ = [‘voit’], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’), (‘fille’, DET, ‘la’)}
RIGHT-ARC(OBJ)	β = [‘.’], σ = [‘fille’, ‘voit’], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’), (‘fille’, DET, ‘la’), (‘voit’, OBJ, ‘fille’)}
REDUCE	β = [‘.’], σ = [‘voit’], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’), (‘fille’, DET, ‘la’), (‘voit’, OBJ, ‘fille’)}
RIGHT-ARC(PONCT)	β = [], σ = [‘.’, ‘voit’], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’), (‘fille’, DET, ‘la’), (‘voit’, OBJ, ‘fille’), (‘voit’, PONCT, ‘.’)}
TERMINATE	A_{final} = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’), (‘fille’, DET, ‘la’), (‘voit’, OBJ, ‘fille’), (‘voit’, PONCT, ‘.’), (ROOT, ROOT, ‘voit’)}

Figure 2.2: Gold ARC-EAGER transition sequence, with corresponding intermediate configurations, for the sentence: “Jean voit souvent la fille.” (“Jean sees often the girl.”)

EAGER is that it has certain information unavailable when making a RIGHT-ARC(l) transition compared to ARC-STANDARD; the latter requires that the potential dependent have all of its own dependents already attached, and these dependents may be informative for certain linguistic phenomena such as PP-attachment (where the potential dependent to be attached is the preposition and its own dependent is the object of the PP).

On the other hand, ARC-EAGER avoids some parsing situations in which the ARC-STANDARD oracle makes implicit attachment decisions without necessarily having access to adequate information. An example of this can be seen for the French sentence “Jean voit Cécile de France” (“Jean sees [the actress] Cécile de France”), with the diverging gold transition sequences for ARC-STANDARD and ARC-EAGER shown in Figure 2.3. The first couple of transitions shown above are shared by



First transitions	**Configuration**
SHIFT, 'Jean' on stack	β = ['Cécile', 'de', 'France', '.']
LEFT-ARC(SUJ), 'voit' → 'Jean'	σ = ['voit']
	A = {('voit', SUJ, 'Jean')}

ARC-STANDARD **transitions**	ARC-EAGER **transitions**
SHIFT, 'Cécile' on stack	RIGHT-ARC(OBJ), 'voit' → 'Cécile'
SHIFT, 'de' on stack	RIGHT-ARC(DEP), 'Cécile' → 'de'
RIGHT-ARC(OBJ), 'de' → 'France'	RIGHT-ARC(OBJ), 'de' → 'France'
RIGHT-ARC(DEP), 'Cécile' → 'de'	REDUCE, 'France' popped
RIGHT-ARC(OBJ), 'voit' → 'Cécile'	REDUCE, 'de' popped
SHIFT, 'voit' on stack	REDUCE, 'Cécile' popped
RIGHT-ARC(PONCT), 'voit' → '.'	RIGHT-ARC(PONCT), 'voit' → '.'
SHIFT, 'voit' on stack	TERMINATED.
TERMINATED.	

Figure 2.3: Intermediate configuration followed by separate transition sequences of ARC-STANDARD and ARC-EAGER parsing for the sentence: "Jean voit Cécile de France." ("Jean sees [the actress] Cécile de France.")

both approaches. Then below on the left is the gold transition sequence for ARC-STANDARD, while below on the right is the gold transition sequence for ARC-EAGER.

As we can see, the ARC-STANDARD oracle performs a SHIFT that pushes 'Cécile' onto the stack, which is necessary because 'Cécile' has the dependent 'de' to the right. This SHIFT can be glossed as "the word form has further dependents on its right". Indeed, if the sentence were instead "Jean voit Cécile de son balcon" ("Jean sees Cécile from his balcony"), the ARC-STANDARD oracle would directly perform an ARC-RIGHT(OBJ) transition because 'Cécile' has no further dependents. During prediction, an oracle uses features that primarily look at two focal positions, the front of the buffer and the top of the stack, so it may be tough to determine whether a word form at the front of the buffer has further dependents to its right. Failing to correctly make this determination is costly for ARC-STANDARD: in the first example 'Cécile' would no longer be able to take 'de' as a dependent, and in the alternate example 'Cécile' would either be forced to incorrectly take 'de' as a dependent or remain stranded on the stack and fail to be accessible as a dependent of 'voit'.

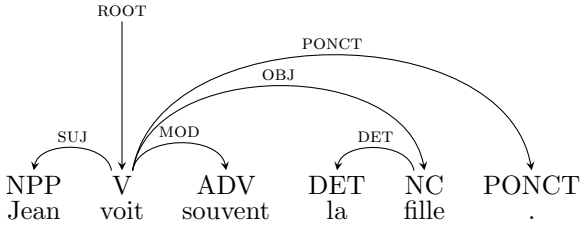
2.2.3 A Multiple-Candidate Variant: ARC-EAGER-MC

We now move on to a presentation of a transition-based parsing implementation that we term ARC-EAGER-MC, and which we have developed to introduce additional context for attachment decisions compared to the existing approaches. This variant arises as a way to address the fact that in the existing approaches transitions either choose a governor or rule out a governor for a word form, but never consider multiple possible governors simultaneously. This approach is a close variant of ARC-EAGER that utilizes a new complex transition that allows only certain sequences of REDUCE transitions followed by an arc-building transition, with REDUCE no longer included as a distinct transition. While one might imagine that this change will result in a transition system that is correct for a reduced class of projective dependency trees compared to the full class associated with ARC-STANDARD and ARC-EAGER, we will show later in this section that ARC-EAGER-MC is actually correct for the same class of graphs.

Let us introduce an extension of the remove operation, $remrep(\cdot, c)$, which applies the $rem(\cdot)$ operation c times on a data structure. In the ARC-EAGER-MC implementation of transition-based dependency parsing framework, the set of transitions used by the transition model contains three types of transitions, with the SHIFT transition operating the same as before for ARC-STANDARD and ARC-EAGER. The following are the set of transitions that move from a previous configuration $c_i = (A_i, \beta_i, \sigma_i)$ to a new configuration $c_j = (A_j, \beta_j, \sigma_j)$:

1. LEFT-ARC(k, l) (for a stack position k and a functional role label l) adds a new arc from the word form at the front of the buffer to the word form at position k in the stack, then pops word forms from the stack $k + 1$ times. Formally, $A_j = A_i \cup \{(\beta_i[0], l, \sigma_i[k])\}$, $\beta_j = \beta_i$, and $\sigma_j = remrep(\sigma_i, k + 1)$. As a precondition for taking this type of transition, the word form at position k on the stack must not be the artificial root note and must be the deepest word form on the stack that still transitively governs the word form on top of the stack. Formally, $\sigma_i[k] \neq w_0$ and $k = \max(u)$ s.t. $\sigma[u] \rightarrow^* \sigma[0]$ holds for A_i.

2. RIGHT-ARC-MC(k, l) (for a stack position k and a functional role label l) adds a new arc from the word form at position k in the stack to the word form at the front of the buffer, then pops word forms from the stack k times, leaving the governor on top of the stack, then moves the word form at the front of the buffer onto the top of the stack. Formally, $A_j = A_i \cup \{(\sigma_i[k], l, \beta_i[0])\}$, $\beta_j = rem(\beta_i)$, and $\sigma_j = add(remrep(\sigma_i, k), \beta_i[0])$. As a precondition for taking this type of transition, the word form at the front of the buffer must not already have a governor, and the word form at position k in the stack must transitively



Transitions	Configurations
INITIALIZE	β = [‘Jean’, ‘voit’, ‘souvent’, ‘la’, ‘fille’, ‘.’], σ = [], A = {}
SHIFT	β = [‘voit’, ‘souvent’, ‘la’, ‘fille’, ‘.’], σ = [‘Jean’], A = {}
LEFT-ARC$(0, \text{SUJ})$	β = [‘voit’, ‘souvent’, ‘la’, ‘fille’, ‘.’], σ = [], A = {(‘voit’, SUJ, ‘Jean’)}
SHIFT	β = [‘souvent’, ‘la’, ‘fille’, ‘.’], σ = [‘voit’], A = {(‘voit’, SUJ, ‘Jean’)}
RIGHT-ARC$(0, \text{MOD})$	β = [‘la’, ‘fille’, ‘.’], σ = [‘souvent’, ‘voit’], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’)}
SHIFT	β = [‘fille’, ‘.’], σ = [‘la’, ‘souvent’, ‘voit’], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’)}
LEFT-ARC$(0, \text{DET})$	β = [‘fille’, ‘.’], σ = [‘souvent’, ‘voit’], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’), (‘fille’, DET, ‘la’)}
RIGHT-ARC$(1, \text{OBJ})$	β = [‘.’], σ = [‘fille’, ‘voit’], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’), (‘fille’, DET, ‘la’), (‘voit’, OBJ, ‘fille’)}
RIGHT-ARC$(1, \text{PONCT})$	β = [], σ = [‘.’, ‘voit’], A = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’), (‘fille’, DET, ‘la’), (‘voit’, OBJ, ‘fille’), (‘voit’, PONCT, ‘.’)}
TERMINATE	A_{final} = {(‘voit’, SUJ, ‘Jean’), (‘voit’, MOD, ‘souvent’), (‘fille’, DET, ‘la’), (‘voit’, OBJ, ‘fille’), (‘voit’, PONCT, ‘.’), (ROOT, ROOT, ‘voit’)}

Figure 2.4: Gold ARC-EAGER-MC transition sequence, with corresponding intermediate configurations, for the sentence: “Jean voit souvent la fille.” (“Jean sees often the girl.”)

govern the word form at the top of the stack. Formally, $(\cdot, \cdot, \beta_i[0]) \notin A_i$, and $\sigma[k] \rightarrow^* \sigma[0]$ holds for A_i.

3. SHIFT moves the word form at the front of the buffer onto the top of the stack. Formally, $A_j = A_i$, $\beta_j = rem(\beta_i)$, and $\sigma_j = add(\sigma_i, \beta_i[0])$. There are no preconditions for taking this type of transition.

For the derivation of a gold transition sequence for a sentence, transitions are selected at each configuration based on an order of descending priority, with arc-building transitions having highest priority as was the case for the other approaches. This time, both LEFT-ARC(k, l) and RIGHT-ARC-MC(k, l) again simply require that the created arc is part of the gold dependency tree for the sentence. Figure 2.4 illustrates the ARC-EAGER-MC gold transition sequence for the same sentence used previously, “Jean voit souvent la fille” (“Jean sees often the girl”), with the gold dependency tree for the sentence shown on top.

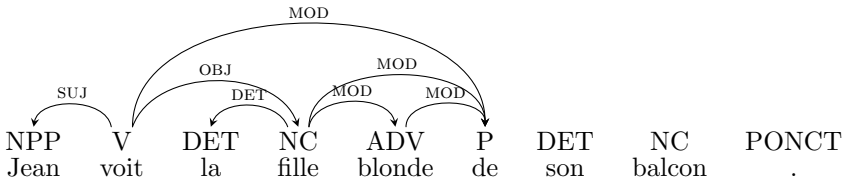
The transition responsible for allowing the consideration of multiple governors is RIGHT-ARC-MC(k, l), which represents a multiple candidate transition set up to directly compare different viable positions k on the stack; specifically, given a possible dependent at the front of the buffer, the oracle decides between multiple potential governor word forms at different positions on the stack. Note that LEFT-ARC(k, l), on the other hand, does not really compare between different positions, as for this transition there is a single viable k in any configuration: the deepest position on the stack whose word form transitively governs the word form on top of the stack given the current set of arcs. Our reason for introducing LEFT-ARC(k, l) is that it allows us to simplify the transition system by eliminating the need for an explicit REDUCE transition.

Why consider multiple candidates?

Our motivation for exploring this multiple-candidate variant of transition-based parsing stems from our desire to use more context in attachment decisions during parsing, as previously mentioned in the lessons we took away from the French benchmarking experiment in Section 2.1.2. ARC-EAGER-MC constitutes our first method of incorporating a framework into parsing in which one can consider multiple candidate governors simultaneously for a given dependent; a subsequent method for achieving this will be presented in Chapter 3, through the use of post-processing parse correction.

To further explain why we believe it may be useful to consider multiple candidate governors simultaneously, consider the configuration represented at the top of Figure 2.5 for a simple French sentence containing a prepositional phrase, "Jean voit la fille blonde de son balcon" ("Jean sees the blonde girl from his balcony"). Below on the left of the figure, we see the sequence of three transitions, two REDUCE followed by a RIGHT-ARC(MOD), that an ARC-EAGER system would have to take in order to correctly process this part of the sentence. Alternatively, below on the right, we see that a single transition RIGHT-ARC-MC$(2, \text{MOD})$ is taken by an ARC-EAGER-MC system to correctly process this part of the sentence.

Depending on the set of features used by the oracle, an ARC-EAGER transition system might not take into account the fact that 'voit' is a possible — and in this case, preferable — governor compared to 'fille' or 'blonde' for the preposition 'de'. The oracle must then essentially make independent and non-retractable decisions as to whether 'blonde', and then 'fille', is the correct governor before properly considering 'voit'. The ARC-EAGER-MC oracle, on the other hand, can simultaneously take into account all of the reasonable candidate governors to the left of the preposition 'de' when making its decision (these candidates appear listed in the stack).



Configuration

β = [‘de’, ‘son’, ‘balcon’, ‘.’]
σ = [‘blonde’, ‘fille’, ‘voit’]
A = {(‘voit’, SUJ, ‘Jean’),
(‘fille’, DET, ‘la’),
(‘voit’, OBJ, ‘fille’),
(‘fille’, MOD, ‘blonde’)}

ARC-EAGER **transitions**	ARC-EAGER-MC **transitions**
REDUCE, ‘blonde’ popped	RIGHT-ARC-MC(2, MOD), ‘voit’ → ‘de’
REDUCE, ‘fille’ popped	...
RIGHT-ARC(MOD), ‘voit’ → ‘de’	
...	

Figure 2.5: Comparison of prepositional phrase treatment by ARC-EAGER and ARC-EAGER-MC for the sentence: “Jean voit la fille blonde de son balcon.” (“Jean sees the blonde girl from his balcony.”). Arc from ‘voit’ to ‘de’ is the correct dependency, while arcs from ‘fille’ to ‘de’ and from ‘blonde’ to ‘de’ show alternative governors implicitly or explicitly considered for ‘de’.

The difference between ARC-EAGER and ARC-EAGER-MC is therefore deeper than it may appear at first, as we have unified two basic transition types into a single larger one for the express purpose of introducing more context into each right-directed attachment decision. From a linguistic perspective, this is very appealing for the treatment of dependents like prepositions of coordination conjunctions, whose governors are often to the left and for which multiple reasonable governors exist and are difficult to disambiguate. The key downside, as we we will soon show, is the $O(n^2)$ time complexity caused by considering multiple candidates. And it should be noted that for many linguistic phenomena a multiple candidate approach does not seem to add much utility; consider dependents whose governors are mainly to the right, such as determiners, and those who tend to have rigid positional requirements with respect to their governors, such as nouns and verbs. There is, however, an appealing solution to this dilemma that we explore in our experiments: a hybrid approach that combines ARC-EAGER with ARC-EAGER-MC and chooses which transition set to consider for a configuration depending on the POS category of the first word form in the buffer, with ARC-EAGER-MC used only in the limited cases where it would be most useful.

Single transition ARC-EAGER-MC	**Transition sequence** ARC-EAGER
LEFT-ARC(k, l)	REDUCE k times, then LEFT-ARC(l)
RIGHT-ARC-MC(k, l)	REDUCE k times, then RIGHT-ARC(l)
SHIFT	SHIFT

Table 2.3: Transition conversion table from ARC-EAGER-MC to ARC-EAGER.

Correctness of ARC-EAGER-MC

We first present a proof of correctness for ARC-EAGER-MC with respect to the class of projective dependency trees. Though the ARC-EAGER-MC variant imposes additional restrictions to its transition sequence as compared to ARC-EAGER, we can actually show that for any sentence and corresponding tree in the class $\mathbb{G}$ of projective dependency trees, ARC-EAGER can produce a transition sequence for that tree if and only if ARC-EAGER-MC can produce one as well. Having done so, we state that ARC-EAGER-MC is correct for $\mathbb{G}$ because we know this to be true of ARC-EAGER.

The first direction of the proof requires that ARC-EAGER can produce a transition sequence for a tree in $\mathbb{G}$ if ARC-EAGER-MC can do so. This direction is rather straightforward, as given any transition sequence produced by ARC-EAGER-MC, we can clearly construct a corresponding transition sequence for ARC-EAGER. Table 2.3 lists which subsequence of transitions in ARC-EAGER corresponds to each transition in ARC-EAGER-MC. When ARC-EAGER-MC takes a LEFT-ARC(k, l) transition, then ARC-EAGER should take k REDUCE transitions followed by a LEFT-ARC(l) transition. When ARC-EAGER-MC takes a RIGHT-ARC-MC(k, l) transition, then ARC-EAGER should take k REDUCE transitions followed by a RIGHT-ARC(l) transition. Finally, when ARC-EAGER-MC takes a SHIFT transition, ARC-EAGER should do the same.

The second direction of the proof states that ARC-EAGER-MC can produce a transition sequence for a tree in $\mathbb{G}$ if ARC-EAGER can do so. This direction is trickier, as there are certain subsequences of transitions that are allowed by ARC-EAGER, specifically those in which a REDUCE is followed by a SHIFT, which have no correlate in the ARC-EAGER-MC system. Luckily, we do not need for there to be a one-to-one correspondence between transition sequences for the two systems; rather, we only need to show that, for any tree G in $\mathbb{G}$, there is at least one ARC-EAGER transition sequence that both derives G and has a corresponding transition sequence in ARC-EAGER-MC. To this end, we can use the unique gold transition sequence resulting from gold oracle parsing, as previously presented in the ARC-EAGER description in Section 2.2.2. First, we note that in ARC-EAGER the REDUCE transition can never complete a transition sequence, recalling that termination requires the buffer to become empty yet the REDUCE transition does not remove anything from the buffer. Next, we note that any sequence of k REDUCE transitions followed by an

arc-building transition can be converted into a corresponding arc-building transition in ARC-EAGER-MC. Now all that remains is to show that a REDUCE is never followed by a SHIFT during gold oracle parsing. We reproduce the priority conditions for SHIFT and REDUCE transitions for ARC-EAGER presented earlier, noting that the arc-building transitions have higher priority:

- Do SHIFT if the word form at the front of the buffer does not have its gold governor to its left. Else do REDUCE if possible. Else do SHIFT.

We can see that in order for a REDUCE transition to be selected for a configuration $c_i = (A_i, \beta_i, \sigma_i)$, it must be the case that $\beta_i[0]$ has its gold governor to its left, and we can then infer that its gold governor is somewhere on the stack σ_i. In the subsequent configuration $c_{i+1} = (A_{i+1}, \beta_{i+1}, \sigma_{i+1})$, we note that $\beta_{i+1}[0] = \beta_i[0]$, which means that it still has its gold governor to its left. Therefore, the only way a SHIFT transition can occur from c_{i+1} is if no arc-building transitions are applicable and the REDUCE transition is not allowed. But this is impossible: if the gold governor of $\beta_j[0]$ is on top of the stack a RIGHT-ARC(l) will be taken, and otherwise a REDUCE will be taken. A REDUCE transition is therefore never followed by a SHIFT transition in the gold oracle sequence for G. □

Computational Complexity of ARC-EAGER-MC

We now prove the computational space and time complexities of ARC-EAGER-MC, with similar approaches to those of Nivre (2008) for ARC-EAGER and ARC-STANDARD.

For space complexity, following the proof of Nivre (2008) for ARC-EAGER and ARC-STANDARD, we first note that only one configuration $c = (A, \beta, \sigma)$ needs to be stored at any given time. Assuming that each single word form vertex can be stored in constant time, and given the fact that the number of arcs in a dependency tree is bounded by the number of corresponding vertices, the space complexity for ARC-EAGER-MC is then $O(n)$. □

For time complexity, we first need to bound the length of the transition sequence $C_{0,m}$ for a sentence $x = w_0 w_1 \ldots w_n$. Again following the proofs for ARC-STANDARD and ARC-EAGER presented by Nivre (2008), we note that the SHIFT and RIGHT-ARC-MC(k, l) transitions both decrease the length of β by 1 and increase the length of σ by at most 1. We also note that the remaining transition, LEFT-ARC(k, l), decreases the depth of σ by at least 1. Finally, we can point to the fact that the initial length of β is n, the initial depth of σ is 1, and additionally we know that σ can never have depth less than 0 and a terminal configuration is reached when β has length 0. From this information, we can conclude that the combined number of transitions is bounded by n.

While Nivre (2008) is able to prove $O(n)$ worst-case time complexity for ARC-STANDARD and ARC-EAGER by additionally assuming that oracle and transition functions can be computed in constant time, we cannot do so due to the more complex nature of our transitions. Specifically, the RIGHT-ARC-MC(k, l) transition requires the oracle to make a decision whose complexity is bounded by the number of viable candidate governors; in the worst case this number can approach n, as one could imagine for a heavily right-branching dependency tree. This means that the time complexity for ARC-EAGER-MC is $O(n^2)$. □

2.3 Parsing Experiments

In this section we describe our first set of transition-based parsing experiments for French over the FTBDep, upon which subsequent chapters build. We include here updated parsing evaluations for the popular ARC-STANDARD and ARC-EAGER approaches, as well as a novel evaluation of the ARC-EAGER-MC approach, both alone and in a hybrid setting with ARC-EAGER that we term HYBRID-EAGER. In Section 2.3.1 we will present our methods and experimental setup, while Section 2.3.2 describes the results of our parsing experiments.

Kernel SVM and Hand-Selected Feature Combinations

Before moving on to the experimental setup, we note that the major difference between these experiments and those of the previous French benchmark described in Section 2.1.2 is our use of SVM with polynomial kernels, as opposed to strictly linear models with hand-selected combinations of features. Remember from Section 1.4.1 that a linear model performs binary or multiclass classification by separating positive and negative instances with a hyperplane in a multidimensional space, where each dimension corresponds to a feature. However, it is clear from the previous works on machine learning approaches to parsing in the NLP literature that at least some non-linear relationships between basic features need to be captured, by which we mean that there exist pairs (or larger combinations) of basic features that need to be considered jointly in order to be truly informative.

In the traditional approach used in the benchmark, which is very popular in the parsing literature, non-linear relationships are captured by adding dimensions corresponding to some hand-selected combinations of basic features. This approach has the advantage of being almost as efficient as basic linear modeling for learning and prediction, as long as the number of additional dimensions is relatively low. The downside is that it is difficult to determine which combinations of features are salient and should be added to the feature space, much more so than it is to

define the basic individual features. Most research that uses this approach seems to select feature combinations in a rather arbitrary fashion, relying in part on linguistic intuitions. Some research goes a bit further and evaluates a number of reasonable sets of additional feature combinations on a development set before settling on a final list, but note that even in this case one cannot hope to approximate an exhaustive search for the most salient set of additional feature combinations.

The polynomial kernel approach, on the other hand, implicitly captures non-linear relations between basic features by learning over an abstract non-linear space consisting of all combinations of features up to a certain length. Prediction remains in the linear space, though it requires the use of a number of support vectors that is bounded by the number of training examples. In our opinion, this is theoretically a preferable approach, as it retains linear time prediction while taking into account all possible combinations of features up to a certain length. We use a cubic kernel, meaning that all feature combinations of length up to three are considered; for comparison, the hand-selected feature combination approach usually includes no more than three basic features per combination. The major downside to the non-linear kernel approach, however, is that in practice the number of support vectors can be rather large, resulting in a slower classification model than that of the alternative hand-selected feature combination approach.

When looking at existing work on transition-based parsing, we note that both SVM and feature combination methods have been used in the literature. The MaltParser software (Nivre et al., 2007b) notably supports both options. Indeed, in the shared task across 13 languages Nivre et al. (2006) used SVM with a quadratic kernel, while in the French benchmarking experiment Candito et al. (2010b) used feature combinations.

Ultimately, we choose to use the kernel SVM approach for classification, under the hypothesis that it will give us the clearest basis for comparing different parsing approaches throughout this thesis; in addition to the parsing experiments in this chapter, subsequent chapters deal with parse correction and new features derived from lexical resources. If we were to use the hand-selected feature combination approach, our fear is that we might unwittingly define feature spaces more favorable to one parser than to another, giving us results that reflect feature engineering quirks rather than fundamental characteristics of the approaches themselves. We note, however, that if we were planning to use one of our parsers in a real world setting, where speed and efficiency are at a premium, it would be worthwhile to use a strictly linear model and make the extra effort to find a good set of hand-selected feature combinations.

2.3.1 Methods and Setup

We now present the setup of our parsing experiments. We describe the following important elements of our methodology: split of the FTBDep into training, development, and test sections; automatic POS tagging and lemmatization of the FTBDep; definition of basic feature templates for each of the three oracle classifiers; SVM learning process; evaluation metrics; tuning of learning parameters; and parser implementation.

FTBDep Split

We divide the FTBDep into the three classic sections for machine learning, using the standard split of Crabbé and Candito (2008) which has been used in multiple works on parsing with both the FTB and FTBDep. The training set, which contains 9,881 of the sentences in the FTBDep (80%), is used to learn optimal oracle transition classifiers for the various transition-based parsing approaches.[1] As mentioned earlier in Section 2.2, each sentence and corresponding gold projective dependency tree must be converted into a gold transition sequence, with features for each transition example defined over its preceding configuration. The development set, which contains 1,235 of the sentences in the FTBDep (10%), is generally used as a preliminary evaluation set to tune learning parameters: based on the performance of a parsing approach using different learning parameter values, an optimal set of values is selected. We will give more details concerning our parameter tuning shortly. Finally, the test set, which contains the remaining 1,235 sentences in the FTBDep (10%), is used for the final evaluation over each parser with optimized learning parameters. The importance of having these three sets, and most crucially the distinction between a training set and a final evaluation test set, is due to the key goal in machine learning of obtaining models that are capable of generalizing well to previously unseen data.

POS Tagging and Lemmatization

In all known approaches for transition-based dependency parsing, the POS categories of word forms in a sentence are crucial features when representing configurations, allowing basic generalization over lexemes. And POS categories are further important for us to identify prior to parsing, as we have noted earlier that in a hybrid approach combining ARC-EAGER with ARC-EAGER-MC the parser will choose which transition set to use depending on the POS category of the word form at the front

[1] Due to some missing annotations, eight sentences are excluded in our experiments leaving a total of 9,873 training sentences.

of the buffer. Of course, when faced with a novel sentence, the POS categories of its word forms are unknown and must be predicted through a separate process, termed *POS tagging*, or as a byproduct of parsing, as is the case for many grammar-based phrase-structure parsing approaches. Similarly, knowing the lemma for a word form is valuable in creating smaller and more generalizable feature spaces, but this information must also be predicted at some point using the corresponding process of *lemmatization*.

From a linguistic perspective, it is important to note that the identification of POS categories for words in a sentence is linked to knowledge about the syntactic structure of a sentence, as some word forms exhibit ambiguity between different POS categories. For instance, the word form 'stop' in English may be a noun or a verb, and its identification as one or the other both informs and is determined by the surrounding syntactic context. For computational dependency parsing approaches, however, it is practical to assume that POS categories of words in a sentence are already known prior to parsing. This assumption holds up reasonably well for languages with lower levels of ambiguity at the morphological level, such as English or French, though it may be less applicable to languages with rich morphology and higher levels of POS ambiguity. For our purposes, we can assume without too much concern that POS tags have already been assigned to word forms in a sentence before parsing.

Regarding the use of automatic prediction of POS categories and lemmas in our experiments, we note that we use automatic methods in spite of the fact that gold POS tags and lemmas are readily available in the FTB. This decision, which is also standard in the dependency parsing literature, is due to the fact that it better approximates real world parsing situations in which absolutely no gold information is available. If one parsing approach were superior to a second one when gold POS tags or lemmas are available but inferior otherwise, we would prefer the second approach.

The process of automatically tagging the FTBDep with POS categories was carried out using the freely-available MElt package (Denis and Sagot, 2009), while automatic lemmatization was carried out using the Lefff lexicon (Sagot, 2010) in concert with heuristics for out-of-vocabulary word forms. Since lemmas are simpler to predict once the POS category of a word form is known, we perform POS tagging prior to lemmatization. Tagging for the FTBDep training set was done differently than for the development and test sets. For the two evaluation sets, a POS tagging model was learned using the full training set and then used to tag all of the sentences in the evaluation sets. In order to have an automatically POS tagged training set for parsing without touching the evaluation sets, we used the *jackknifing* technique: the

Feature Templates		ARC-STANDARD	ARC-EAGER	ARC-EAGER-MC
Buffer:	$\text{LEM}_{\beta[0]}$	×	×	×
	$\text{POS}_{\beta[0]}$	×	×	×
	$\text{LPOS}_{\beta[0]}$	×	×	×
	$\text{LLAB}_{\beta[0]}$	×	×	×
	$\text{RPOS}_{\beta[0]}$	×		
	$\text{RLAB}_{\beta[0]}$	×		
	$\text{LEM}_{\beta[1]}$	×	×	×
	$\text{POS}_{\beta[1]}$	×	×	×
	$\text{POS}_{\beta[2]}$	×	×	×
	$\text{POS}_{\beta[3]}$	×	×	×
Stack:	$\text{LEM}_{\sigma[0]}$	×	×	
	$\text{POS}_{\sigma[0]}$	×	×	
	$\text{LAB}_{\sigma[0]}$	×	×	
	$\text{HLEM}_{\sigma[0]}$	×	×	
	$\text{HPOS}_{\sigma[0]}$	×	×	
	$\text{LPOS}_{\sigma[0]}$	×	×	
	$\text{LLAB}_{\sigma[0]}$	×	×	
	$\text{RPOS}_{\sigma[0]}$	×	×	
	$\text{RLAB}_{\sigma[0]}$	×	×	
	$\text{POS}_{\sigma[1]}$	×	×	
	$\text{POS}_{\sigma[2]}$	×	×	
k-Stack:	$\text{LEM}_{\sigma[k]}$			×
	$\text{POS}_{\sigma[k]}$			×
	$\text{LAB}_{\sigma[k]}$			×
	$\text{HLEM}_{\sigma[k]}$			×
	$\text{HPOS}_{\sigma[k]}$			×
	$\text{LPOS}_{\sigma[k]}$			×
	$\text{LLAB}_{\sigma[k]}$			×
	$\text{RPOS}_{\sigma[k]}$			×
	$\text{RLAB}_{\sigma[k]}$			×
MC-rank:	$\text{NDEPS}_{\sigma[k]}$			×
	$\text{DEPTH}_{\sigma[k]}$			×
	$\text{DIST}_{\sigma[k],\beta[0]}$			×
	$\text{PUNC}_{\sigma[k],\beta[0]}$			×

Table 2.4: Basic feature templates for ARC-STANDARD, ARC-EAGER and ARC-EAGER-MC.

training set was split into ten parts, and then ten separate iterations were performed with a different held out set to be tagged with a model trained on the other nine. Finally, automatic lemmatization was performed uniformly over the entire FTBDep.

Feature Templates

Table 2.4 lists the basic feature templates for each of the three transition oracles corresponding to ARC-STANDARD, ARC-EAGER and ARC-EAGER-MC. The notation

convention we use includes shorthand terms for the type of information over one or more word forms, with subscripts indicating their locations in the current configuration. For instance, $\text{LEM}_{\beta[0]}$ indicates the lemma of the word form at the front of the buffer. The primary shorthand terms are LEM for lemma, POS for part-of-speech, and LAB for the functional role label in which the subscripted word form is the dependent. The prefix H indicates information concerning the governor (if one currently exists) of a subscripted word form, and the prefixes L and R indicate information concerning the farthest left and right dependent (if one currently exists) of a subscripted word form, respectively. Finally, NDEPS indicates the number of dependents, DEPTH indicates a binned value of k, DIST indicates a binned linear distance between two word forms, and PUNC indicates whether there is a punctuation mark linearly between two word forms.

The features over the buffer are mostly the same for the three transition systems, with the notable exception of two features — $\text{RPOS}_{\beta[0]}$ and $\text{RLAB}_{\beta[0]}$ — involving the farthest right dependent of the word form at the front of the buffer. These two features are only relevant for ARC-STANDARD, since the eager approaches never return a word form from the stack to the buffer, meaning that the word form at the front of the buffer never has any dependents to its right during parsing. Features involving the stack are divided into a basic set, used by ARC-STANDARD and ARC-EAGER, defined over the first few word forms at the top of the stack, and a k-sensitive set, used by ARC-EAGER-MC for its arc-building transitions and defined over the word form at position k on the stack. Finally, we note that the four feature templates listed under MC-rank are used for ranking between candidates at different values of k exclusively for the $\text{RIGHT-ARC-MC}(k, l)$ transition.

SVM Learning

As previously explained in Section 1.4, there are two main machine learning approaches that we use to optimize the oracle transition function for our transition systems: multiclass SVM and ranking SVM, each using a polynomial kernel of degree d=3. Also, we note that SVM model training and prediction was performed using the LIBSVM package (Chang and Lin, 2011) for multiclass models, and the SVM^{light} package (Joachims, 1999) for ranking models.

For the ARC-STANDARD and ARC-EAGER transition oracles, multiclass SVM is adequate by itself because any given configuration can be represented using a single feature vector. We thus use multiclass SVM to learn three classifiers that jointly represent the oracle. The first classifier decides which of the basic transition types to use, setting aside the label l. If $\text{LEFT-ARC}(l)$ is chosen, then l is chosen by a separate left-labeling classifier; similarly, if $\text{RIGHT-ARC}(l)$ is chosen, then l is chosen

by a separate right-labeling classifier.

For the ARC-EAGER-MC transition oracle, we make use of both multiclass and ranking SVM approaches to learn four models that jointly represent the oracle. As before, the first classifier decides which of the basic transition types to use, this time setting aside both the label l and the position k. Note that in the previous section we defined feature templates for ARC-EAGER-MC over a particular position k. Since this position is unknown for the first classifier, we simply consider these categorical features to be simultaneously 'true' for values found at different valid k positions; for instance, if 0 and 1 are both valid k positions, and the POS categories of $\sigma[0]$ and $\sigma[1]$ are N and V, both indicator features $\text{LEM}_{\sigma[k]} = \text{N}$ and $\text{LEM}_{\sigma[k]} = \text{V}$ will fire. If LEFT-ARC$(\cdot, l)$ is chosen, then k is automatically determined to be the deepest position that transitively governs $\sigma[0]$, and l is chosen by a separate left-labeling classifier. If RIGHT-ARC-MC$(\cdot, l)$ is chosen, then a separate feature vector is constructed for each valid k position, and a ranking model is used to select the best k; subsequently, l is chosen by a separate right-labeling classifier using the feature vector corresponding to the best k.

Finally, the HYBRID-EAGER transition oracle is not actually trained separately. Rather, during parsing it simply calls on the ARC-EAGER and ARC-EAGER-MC oracles to predict transitions between configurations. The policy we use is as follows: the ARC-EAGER-MC oracle is called for configurations in which the word form at the front of the buffer is a preposition or a conjunction, while the ARC-EAGER oracle is called for all other configurations.

Evaluation Metrics

In order to evaluate our parsing approaches, we use simple, straightforward metrics that are widely accepted in the dependency parsing community. The first is *unlabeled attachment score (UAS)*, which ignores the arc labels and simply determines the percentage of word forms that have been either assigned the correct governor or correctly identified as the root. The second is *labeled attachment score (LAS)*, which is a stricter metric that additionally requires that the governing arc for a word form have the correct functional role label. Given the fact that transition-based parsers, as we have defined them, will always output complete dependency trees, we note that each word form will always be assigned exactly one governing arc (with the dummy root as a possible governor). This is a nice property for evaluation, saving us the need to deal with the notions of precision, recall, and f-measure.

An additional point concerning our evaluation is that we do not score punctuation mark dependents, following the standard methodology in the statistical parsing literature. While punctuation marks are treated as word forms for the purpose of

parsing a sentence, we note that punctuation attachments mostly carry little meaning and are often annotated inconsistently, as we have found to be the case for the FTBDep. When calculating LAS or UAS, we choose to only score the correct assignation of a governor to a dependent if the dependent is not a punctuation mark.

Finally, in order to determine the statistical significance of differences in LAS or UAS between two parsing approaches, we choose to use McNemar's statistical test, which assesses the significance of the difference between two correlated proportions, such as might be found in the case where the two proportions are based on the same sample of subjects. In our case, the two proportions correspond to attachment score results for two parsing approaches, with the "subjects" being word forms in the FTBDep test set that are assigned either the correct or an incorrect governor. Differences in performance are tested for significance at the level p=0.05.

Tuning Learning Parameters

Before performing the final evaluation on the FTBDep test set, we tuned some of the LIBSVM and SVM^{light} learning parameters using the FTBDep development set. The parameter values leading to the highest LAS and UAS over sentences in the development set were identified and locked in for the final evaluation over the test set.

The learning parameters we fixed from the start were the use of the C-SVC learning type for SVM, as described in Section 1.4.2, as well as the use of a polynomial kernel with d=3. The inner coefficient of s=0.1 for the polynomial kernel (cf. Equation 1.6), was found through development set tuning. In addition, after tuning we settled on a C-SVC regularization parameter with value C=1 for multiclass models and C=0.05 for ranking models, and a C-SVC termination criterion parameter with value ϵ=1.0 for both multiclass and ranking models. Finally, we decided to use no class bias for any of our models.

Another choice we made during development was to split each model even further depending on the POS category of the word form at the front of the buffer in a configuration. The POS groupings are listed in Table 2.5. This was done for reasons of computational efficiency, as it significantly reduces the overall time and space required for training and parsing. This type of model splitting is a standard method used in transition-based dependency parsing experiments (Nivre et al., 2006; Candito et al., 2010b).

Finally, we tested two HYBRID-EAGER approaches, based on the low UAS results for coordinating conjunctions on the development set when using the standard hybrid approach. The standard approach that we have discussed previously in this chapter, HYBRID-EAGER-1, uses ARC-EAGER-MC transitions when the word form at

Model Group	Fine POS Categories
Adjectives	ADJ, ADJWH
Adverbs	ADV, ADVWH
Conjunctions	CC, CS
Clitics	CLO, CLR, CLS
Determiners	DET, DETWH
Foreign Words	ET
Interjections	I
Nouns	NC, NPP
Prepositions	P,P+D, P+PRO
Punctuation	PONCT
Prefixes	PREF
Pronouns	PRO, PROREL, PROWH
Verbs	V, VIMP, VINF, VPP, VPR, VS

Table 2.5: Grouping of classification and ranking models by fine POS category of word form at the front of the buffer in transition-based parsing experiments.

the front of the buffer is either a preposition or a conjunction and ARC-EAGER transitions otherwise. We considered an alternate hybrid approach, HYBRID-EAGER-2, which further restricts the use of ARC-EAGER-MC transitions to only prepositions.

Parser Implementation

We use our own parser implementation, coded with Python (Bird et al., 2009), for the experiments in this chapter as well as for all subsequent chapters. We plan to release software implementing all of the methods from this thesis in the near future.

As we have noted before, a freely available, high quality and flexible implementation of transition-based parsing already exists in MaltParser (Nivre et al., 2007b). The reasoning behind using our own code is that it allowed us to implement our ARC-EAGER-MC variant in a straight-forward manner, and also provided us with a code base that was easily extended to implement later methods in our thesis like the parse correction algorithm described in Chapter 3.

2.3.2 Results

This section describes the results of our transition-based parsing experiments, with special attention paid to the multiple-candidate variant for difficult attachment types. We also briefly consider the running times for training and parsing under different transition systems, which gives us a sense of the tradeoffs between speed and accuracy in transition-based parsing. Table 2.6 shows the evaluation results for the four transition systems over the FTBDep test set, with LAS and UAS results

Transition System	Overall		Preps	Coords
	LAS	UAS	UAS	UAS
ARC-STANDARD	87.3	89.9	83.8	59.2
ARC-EAGER	87.1	89.7	84.0	64.4
ARC-EAGER-MC	86.5	89.5	84.6	65.3
HYBRID-EAGER-1	87.1	89.7	84.5	62.8
HYBRID-EAGER-2	87.2	89.8	84.5	64.3

Table 2.6: LAS and UAS results, in percent, over the FTBDep test set for the five evaluated transition systems: ARC-STANDARD, ARC-EAGER, ARC-EAGER-MC, HYBRID-EAGER-1 for prepositions and conjunctions, and HYBRID-EAGER-2 for only prepositions. Also includes UAS results when restricting scoring dependents to prepositions (P,P+D) or coordinating conjunctions (CC).

over all word forms as well as UAS results when considering difficult attachment types (those with either preposition or coordinating conjunction dependents). Scoring word forms in the test set, which do not include punctuation marks, amount to 31,404 total word forms, with 5,706 prepositions and 801 coordinating conjunctions.

Original Transition Sets

When considering the original transition sets, we note first of all that we obtain results comparable to and even slightly better than those of the previous benchmarking evaluation for French discussed in Section 2.1.2. Specifically, the LAS and UAS for ARC-EAGER on the FTBDep test set when using MaltParser with hand-built feature combinations (86.7 LAS, 89.3 UAS) is slightly lower than it is for our SVM implementation with a cubic kernel (87.1 LAS, 89.7 UAS).

A second observation concerns the relative performance of ARC-STANDARD and ARC-EAGER. The performance of ARC-STANDARD is better in terms of overall LAS and UAS, though the difference is not statistically significant. On the other hand, ARC-EAGER has a better UAS for both prepositions and coordinating conjunctions. Of these differences, the one concerning coordinating conjunctions is statistically significant. The better performance of ARC-EAGER for difficult attachment types is an interesting result, and consistent with our observations in Section 2.2.2 when comparing the two approaches: ARC-STANDARD requires the parser to make early implicit decisions concerning right arcs, and this could be a problem for ambiguous, predominantly right-directed attachments. Remember that for any configuration during ARC-STANDARD parsing, if a RIGHT-ARC(l) transition is taken, then the word form at the top of the stack can no longer take additional dependents; conversely, if a SHIFT transition is taken in order to allow for additional dependents on the right, but there turn out to be none, then the shifted word form is stuck and can no longer

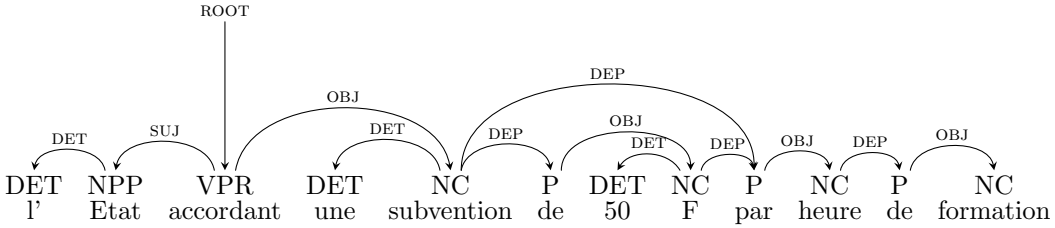


Figure 2.6: Comparison of the incorrect ARC-STANDARD (with arc from 'subvention' to 'par') and correct ARC-EAGER (with arc from 'F' to 'par') parse trees for the partial French sentence: "...l'Etat accordant une subvention de 50 F par heure de formation ..." ("...the state according an allowance of 50 F per hour of training ...")

take a governor on its left.

An example from the FTBDep development set that may support this observation is shown in Figure 2.6, with predicted parses for the partial French sentence "...l'Etat accordant une subvention de 50 F par heure de formation ..." ("...the state according an allowance of 50 F per hour of training ...") under ARC-STANDARD and ARC-EAGER. Consider the configuration during parsing where the noun 'F' is at the front of the buffer, after having taken '50' as a dependent on its left. Under ARC-STANDARD, the oracle must essentially decide whether to attach 'F' as an object dependent of 'de' immediately, or wait for additional dependents on its right; if it waits but no additional dependents are forthcoming, then it will no longer be able to make the clearly correct attachment of 'F' as an object of 'de'. Therefore, ARC-STANDARD must implicitly decide whether or not 'par' is a dependent of 'F' before it has these two word forms in the focus positions (top of the stack and front of the buffer). This may be the reason that ARC-STANDARD makes an incorrect determination and chooses to immediately attach 'F' to 'de', removing 'F' from further consideration as a governor. On the other hand, ARC-EAGER is able to take the immediate transition that attaches 'F' as an object of 'de', moving 'par' to a focus position at the front of the buffer that allows the oracle classifier to better predict its governor.

Novel Transition Sets

Now we turn to results concerning the novel transition sets introduced in this chapter, ARC-EAGER-MC and the two HYBRID-EAGER approaches. For ARC-EAGER-MC, we see that the overall LAS and UAS are substantially lower than those for ARC-STANDARD and for ARC-EAGER, with the difference being statistically significant; this is a disappointing result. On the other hand, the UAS for prepositions and for coordinating conjunctions is higher for ARC-EAGER-MC, with this difference being

statistically significant for both prepositions and coordinating conjunctions when compared to ARC-STANDARD, and nearly significant (at p=0.08) for prepositions when compared to ARC-EAGER. This observation is consistent with our hypothesis, presented in Section 2.2.3, that the use of multiple-candidate arc-building transitions is theoretically useful for treating ambiguous right-directed dependencies, as is the case for PP-attachment and coordination, while not necessarily being suited to other types of dependencies.

Following along our discussion in Section 2.2.3, the HYBRID-EAGER setting was the proposed solution to address the problem of multiple-candidate transitions being suitable for only certain types of attachment. By applying ARC-EAGER-MC transitions only when a potential right-dependent (the word form at the front of the buffer) is a preposition or a conjunction and using ARC-EAGER transitions otherwise, we would hopefully produce an oracle that combines the benefits of both approaches. The setting HYBRID-EAGER-1 uses multiple candidate transitions for both prepositions and conjunctions, with the expected result that overall LAS and UAS are similar to ARC-EAGER with an improved UAS for prepositions comparable to that achieved by ARC-EAGER-MC. Unexpectedly, however, UAS for coordinating conjunctions is lower than that of ARC-EAGER-MC and even of ARC-EAGER. This negative result for coordinating conjunctions had already been observed in the development set, leading to the setting HYBRID-EAGER-2 that only uses sc arc-eager-mc transitions for prepositions. This setting succeeds in achieving the desired result of combining the advantages of ARC-EAGER and ARC-EAGER-MC: it produces overall LAS and UAS results that are slightly higher than either of those approaches, a preposition UAS comparable to the high one for ARC-EAGER-MC, and additionally a coordinating conjunction UAS comparable to the high one for ARC-EAGER.

Running Time

Earlier in this section we explained our reasoning for using the slower polynomial kernel SVM approach, compared to the strictly linear modeling approach with hand-built feature combinations, in order to better compare parsing approaches and avoid spurious results dependent more on feature combination engineering than inherent characteristics of the different approaches. Although the parsers we test in this and later chapters take substantially longer to run than those that can be found in more optimized implementations, such as MaltParser (Nivre et al., 2007b), it is still useful to compare the relative running times within our set of parsing approaches.

Table 2.7 shows the running times of parsers using the five transition system settings over the FTBDep test set, in minutes and seconds. The ARC-STANDARD and ARC-EAGER parsers are both relatively fast, taking around three minutes each

Transition System	Running Time
ARC-STANDARD	3:11
ARC-EAGER	3:10
ARC-EAGER-MC	9:54
HYBRID-EAGER-1	6:26
HYBRID-EAGER-2	6:08

Table 2.7: Running times (min:sec) over the FTBDep test set for the following evaluated transition systems: ARC-STANDARD, ARC-EAGER, ARC-EAGER-MC, HYBRID-EAGER-1 for prepositions and conjunctions, and HYBRID-EAGER-2 for only prepositions.

POS Category	Frequency	UAS
Overall	31,488	89.1
NC	7,853	91.5
DET	5,137	98.0
P	4,748	82.3
ADJ	2,700	93.1
V	2,018	87.8
ADV	1,545	83.2
NPP	1,493	89.4
P+D	1,145	88.4
VPP	1,108	80.9
VINF	812	93.6
CC	783	63.6
CLS	448	98.9
CS	366	72.4
PROREL	341	94.1
CLR	236	98.3
PRO	209	79.9
CLO	178	92.7
VPR	168	72.6

Table 2.8: Breakdown of UAS results by fine POS category of the dependent for the ARC-EAGER transition system on the FTBDep development set. POS categories are listed in descending order of frequency, with only those POS with at least 100 occurrences in the FTBDep development set being included in the table.

to parse the FTBDep test set. As expected, ARC-EAGER-MC, which uses a multiple-candidate transition set for all configurations, is the slowest at around ten minutes. Finally HYBRID-EAGER-1 and HYBRID-EAGER-2, the two compromise settings that use the ARC-EAGER-MC transition set when the word from at the front of the stack is a preposition (both settings) or a conjunction (HYBRID-EAGER-1 only), take six and a half minutes and six minutes, respectively.

In light of the halving of parsing speed when using ARC-EAGER-MC transitions, even for the HYBRID-EAGER approaches, the minor increases in parsing accuracy

are disappointing. We do note that the increase in time complexity for multiple-candidate approaches is from linear to quadratic, which is not as severe as the jump from linear to at least cubic that is suffered when using global graph-based dependency parsing approaches. Nevertheless, the minor increase in parsing accuracy gained when using a multiple-candidate approach does not seem to sufficiently offset the rise in time complexity when compared to the simpler ARC-EAGER transition set.

Difficulty of Attachment for Different POS Categories

From Table 2.6, it is clear that preposition and coordinating conjunction dependents are more difficult to attach than average, as the preposition UAS and coordinating conjunction UAS are lower than the overall UAS; this is true across all of the tested parsing approaches. In order to evaluate the relative difficulty in attaching all of the different POS categories, we provide in Table 2.8 full UAS results broken down by POS category for the representative ARC-EAGER transition system on the FTBDep development set, with only the 18 POS categories that occur at least 100 times being included for clarity (out of the 28 total POS categories).

We observe that two of the most prominent sources of error can be attributed to coordination and PP-attachment. Coordination (with CC dependent) accounts for around 10% of incorrect attachments and has an error rate of nearly 40%, while PP-attachment (with P or P+D dependent) accounts for around 30% of incorrect attachments and has an error rate over 15%. Other POS categories of note that have well below average UAS include adverbs (ADV), verb past participles (VPP), and subordinating conjunctions (CS); it would be interesting to investigate further why those POS are difficult to attach. In the remainder of our thesis, however, we choose to focus our attention on improving the treatment of coordination and PP-attachment.

Chapter 3

Efficient Large-Context Parse Correction

As for the right way, the correct way, and the only way, it does not exist.

— *Nietzsche*

Having presented the base approaches to transition-based parsing used in this thesis, we now continue the exploration that we began in the previous chapter into the first main thread of our thesis: the search for a framework in which additional syntactic context can be considered for attachment decisions. In the previous chapter, our approach was to incorporate directly into the parser's transition system a way to consider multiple candidate governors simultaneously for a given dependent. While that approach showed some promise, particularly for the ambiguous linguistic phenomena of PP-attachment and coordination, the fact that it increased the computational complexity of transition-based parsing from linear to quadratic was a considerable downside. In this chapter, we stick with a traditional transition system and instead seek to incorporate additional syntactic context through the use of two-stage parsing approach: a first-stage transition-based parsing pass is followed by a second-stage *parse correction* pass that reconsiders attachments with the dual benefits of more surrounding syntactic context as well as the ability to simultaneously compare multiple candidate governors for a dependent efficiently.

The presentation of this chapter closely follows one of our previously published works (Henestroza Anguiano and Candito, 2011), with a notable difference being that we update its machine learning algorithms in order to bring it in line with the polynomial kernel SVM approach used throughout this thesis. We also include here a novel investigation into self-training for two-stage transition-based dependency parsing, as well as an application to domain adaptation for parsing.

In Section 3.1 we present the problem of learning to correct dependency parse errors, describing both the neighborhood correction framework first used by Hall and Novák (2005) as well as our innovations in using a ranking learner and accessing larger syntactic context.

In Section 3.2 we discuss an interesting self-training approach to learning with partially unannotated data in a two-stage parsing system, which has previously been used successfully by McClosky et al. (2006) with a phrase-structure parser coupled with a reranker. We investigate a corresponding approach for dependency parsing that involves a transition-based parser coupled with a corrector, and also note how this can be applied to the problem of domain adaptation.

Finally, Section 3.3 includes correction experiments for French in which we compare the performance of baseline ARC-STANDARD and ARC-EAGER parsers from Chapter 2 to two-stage systems that additionally correct dependency errors from the parsing stage, with special attention given to the difficult attachment types of PP-attachment and coordination. We also evaluate to what extent the self-training approach is effective for two-stage dependency parsing, both in-domain and out-of-domain.

3.1 Learning to Correct Parse Errors

In the approaches we consider for syntactic dependency parse correction, attachments in an input parse tree are revised by selecting, for a given dependent, the best governor from within a small set of candidates. The motivation behind parse correction is that attachment decisions, especially difficult ones like PP-attachment and coordination, may require substantial contextual information in order to be made accurately. Because syntactic dependency parsers predict the parse tree for an entire sentence, they may not be able to take into account sufficient context when making attachment decisions, due to computational complexity. Assuming nonetheless that a predicted parse tree is mostly accurate, parse correction can revise difficult attachments by using the predicted tree's syntactic structure to restrict the set of candidate governors and extract a rich set of features to help select among them. Parse correction is also appealing because it is *parser-agnostic*: it can be trained to correct the output of any dependency parser. However, in this chapter we will test parse correction exclusively over trees predicted by the transition-based parsers investigated earlier in Chapter 2.

We organize our discussion as follows: We start off in Section 3.1.1 with a presentation of related work concerning both parse correction and past approaches for resolving ambiguity in PP-attachment and coordination. Section 3.1.2 then introduces the neighborhood correction framework we adopt in our investigation, which is based on the work of Hall and Novák (2005). Finally, in Section 3.1.3 we present our large-context innovations in implementing the framework, which include the use of a ranking learner, the introduction of specialized features for treating difficult attachments, and access to more feature combinations through SVM with a polynomial kernel.

3.1.1 Related Work

Previous research directly concerning parse correction includes that of Attardi and Ciaramita (2007), working on English and Swedish, who use an approach that considers a fixed set of revision rules: each rule describes movements in the parse tree leading from a dependent's original governor to a new governor, and a classifier is trained to select the correct revision rule for a given dependent. One drawback of this approach is that the classes lack semantic coherence: a sequence of movements does not necessarily have the same meaning across different syntactic trees. Hall and Novák (2005), working on Czech, define a neighborhood of candidate governors centered around the original governor of a dependent, and a Maximum Entropy model determines the probability of each candidate-dependent attachment. We follow pri-

marily from their work in our use of neighborhoods to delimit the set of candidate governors. As we will elaborate upon later, our main innovations over this approach are: (i) the use of specialized corrective features designed for difficult attachment types (coordination and PP-attachment); (ii) a ranking model trained directly to select the true governor from among a set of candidates; and (iii) non-linear feature combinations, crucial for making difficult attachment decisions and achieved using SVM learning with polynomial kernels.

There has also been other work on techniques similar to parse correction. Attardi and Dell'Orletta (2009) investigate *reverse revision*: a left-to-right transition-based model is first used to parse a sentence, then a right-to-left transition-based model is run with additional features taken from the left-to-right model's predicted parse. This approach leads to improved parsing results on a number of languages. While their approach is similar to parse correction in that it uses a predicted parse to inform a subsequent processing step, this information is used to improve a second parser rather than a model for correcting errors. McDonald and Pereira (2006) consider a method for recovering non-projective attachments from a graph representation of a sentence, in which an optimal projective parse tree has been identified. The parse tree's edges are allowed to be rearranged in ways that introduce non-projectivity in order to increase its overall score. This rearrangement approach resembles parse correction because it is a second step that can revise attachments made in the first step, but it differs in a number of ways: it is dependent on a graph-based parsing approach, it does not model errors made by the parser, and it only outputs non-projective parses.

As a process that revises the output of a syntactic parser, parse reranking is also similar to parse correction. A well-studied area of research, with seminal works using phrase-structure parsing for English (Charniak and Johnson, 2005; Collins and Koo, 2005) and a recent work for French (Le Roux et al., 2011), parse reranking is concerned with reordering n-best ranked trees output by a parser. Parse correction has some advantages compared to reranking: it can be used with parsers that do not produce n-best ranked trees, it can be restricted to specific attachment types, and its output space is not limited to parses appearing in an n-best list. However, reranking has the advantage of selecting a globally optimal parse for a sentence from an n-best list, while parse correction makes locally optimal revisions within the input predicted parse.

Difficult Attachment Types

Alhough we have already previously discussed the importance of correctly treating difficult attachment types (cf. Chapter 2, Section 2.1.2), notably PP-attachment

and coordination, these attachments take center stage in our experiments with parse correction. We therefore believe it necessary to discuss here related work that focuses specifically on the resolution of these attachment types.

Research on PP-attachment traditionally formulates the problem in isolation (Hindle and Rooth, 1993; Pantel and Lin, 2000; Olteanu and Moldovan, 2005). Examples consist of tuples of the form (v, n_1, p, n_2), where either v or n_1 is the true governor of the prepositional phrase containing p and n_2, and the task is to choose between v and n_1. Subsequently, Atterer and Schütze (2007) criticized this formulation as unrealistic because it uses a gold oracle to select candidate governors, and they found that successful approaches for the isolated problem perform no better than state-of-the-art parsers on PP-attachment when evaluated on full sentences. With parse correction, though, candidate governors are identified automatically with no artificial restriction of the form (v, n_1, p, n_2). Though the problem of choosing between a verb and its nominal object as the governor of a PP arises often during correction, so do other scenarios (e.g. choosing between more than two candidate governors).

Research on coordination resolution has also often formulated the problem in isolation. Resnik (1999) uses semantic similarity to resolve noun-phrase coordination of the form (n_1, cc, n_2, n_3), where the coordinating conjunction cc coordinates either the heads n_1 and n_2 or the heads n_1 and n_3. The same criticism as the one made by Atterer and Schütze (2007) for PP-attachment might be applied to this approach to coordination resolution. In another formulation, the input consists of a raw sentence, and coordination structure is then detected and disambiguated using discriminative learning models (Shimbo and Hara, 2007) or coordination-specific parsers (Hara et al., 2009). Finally, other work has focused on introducing specialized features for coordination into existing syntactic parsing models (Hogan, 2007). We note that our approach is again novel with respect to previous work because it directly models the correction of coordination errors made by dependency parsers.

3.1.2 The Neighborhood Correction Framework

The neighborhood correction framework for two-stage dependency parsing, first introduced by Hall and Novák (2005), consists of a process in which attachments from the parse tree of a sentence predicted by a first-stage parser are corrected by considering a neighborhood of alternative candidate governors for each dependent. While their work focused on the recovery of non-projective dependencies after parsing a sentence with a strictly projective parser, in our case we specifically want to preserve projectivity in the final output parse. The main divergence in our formulations is thus that we use an iterative correction process, which facilitates our ability to

INPUT: Predicted parse tree G
LOOP: For each chosen dependent $d \in D$
- Identify candidates C_d from G
- Predict $(\hat{c}, l) = o(G, d, C_d)$
- Update arcs in G: remove $(\cdot, \cdot, d)$, then add $(\hat{c}, l, d)$

OUTPUT: Corrected version of parse tree G

Figure 3.1: The generalized parse correction framework.

ensure projectivity after every correction; Hall and Novák (2005) were comfortable with predicting corrections simultaneously, since they could use a conflict resolution step at the end to ensure basic tree properties.

Formally, our version of the algorithm for parse correction takes as input the predicted parse G of a sentence, and from G a set D of dependent word forms are chosen for attachment revision. For each $d \in D$ in left-to-right sentence order, a set C_d of candidate governors from G is identified, and then the best $\hat{c} \in C_d$, using an oracle $o : (G, d, C_d) \rightarrow (\hat{c}, l)$, is assigned as the new governor of d in G, with the dependency arc labeled as l. Pseudo-code for parse correction is shown in Figure 3.1.

Given that our correction process is iterative, we are guaranteed that the final output parse is a projective tree so long as we can ensure that each correction results in a projective tree. To ensure this we place two key restrictions on revisions: (i) we never revise a direct dependent of the ROOT vertex, so as to ensure connectedness; and (ii) we restrict the set of candidate governors for a dependent to those for which the change in attachment would preserve projectivity.[1]

Choosing Dependents to Revise

Setting aside the direct dependents of the ROOT vertex, which we never revise, different criteria may be used to choose the set D of dependents to revise. In the work of Hall and Novák (2005) and of Attardi and Ciaramita (2007), D contains all vertices in the input parse tree. However, an advantage of parse correction is its ability to focus on specific attachment types, so it may be advantageous to look separately at those dependents that correspond to difficult attachment types like coordination and PP-attachment. As we noted in our evaluation of transition-based parsers for French (cf. Chapter 2, Section 2.3.2), these were two of the largest sources of parsing errors: coordination (with CC dependent) accounted for around 10% of

[1] In our experiments we actually loosen the projectivity restriction for punctuation mark dependents. Since they are unreliably annotated and not scored in our evaluation, we simply ensure that the parse tree would remain projective if dependencies with punctuation mark dependents were ignored.

incorrect attachments and had an error rate of nearly 40%, while PP-attachment (with P or P+D dependent) accounted for around 30% of incorrect attachments and had an error rate over 15%.

Recall that in the representation of coordination for the FTBDep annotation scheme, the first conjunct governs subsequent CCs, with other conjuncts depending on the CC directly on their left. Errors for coordination occur mainly on dependencies in which the CC is a dependent rather than those in which the CC is a governor; indeed, the former type of dependency is the one we track in our evaluations of coordination parsing accuracy. This means that coordination can be corrected primarily by revising those dependencies in which a CC is the dependent. As for the representation of PP-attachment in the FTBDep annotation scheme, we reiterate that PPs are headed by the preposition that introduces them, so this type of attachment can be corrected by revising those dependencies in which a preposition (P or P+D) is the dependent.

Identifying Candidate Governors

The set of candidate governors C_d for a dependent d can be chosen in different ways, of course taking into account the requirement that its selection as the new governor for d conserve projectivity in the parse tree. One method is, for a given d, to let every other word form in G be a candidate governor for d. However, parser error analysis has shown that errors often occur in local contexts.

Parse correction therefore uses a local *neighborhood* of candidate governors, which Hall and Novák (2005) define as a set of vertices $N_m(d)$ around the original predicted governor c_o of d, where $N_m(d)$ includes all word forms in the parse tree G within graph distance m of d with a path that passes through c_o. Note that this excludes any word forms transitively governed by d, which prevents the introduction of cycles in a correct graph. Hall and Novák (2005) find that around 2/3 of the incorrect attachments in the output of Czech parses can be corrected by selecting the best governor from within $N_3(d)$. Similarly, in gold oracle experiments reported in Section 3.3, we find that around 1/2 of coordination and PP-attachments in the output of French parses can be corrected by selecting the best governor from within $N_3(d)$. Figure 3.2 shows a neighborhood of candidate governors for an incorrect PP-attachment from first-stage parsing in the French sentence "Jean voit la fille blonde de son balcon" ("Jean sees the blonde girl from his balcony").

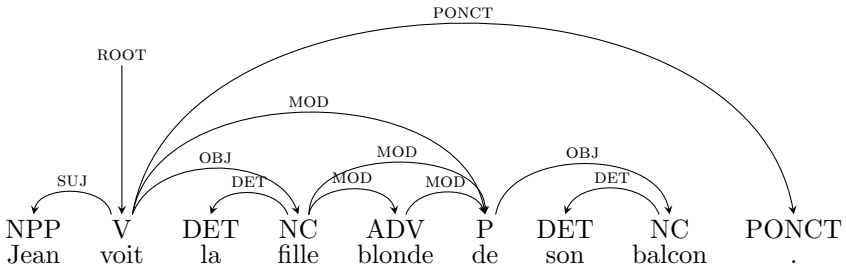


Figure 3.2: Neighborhood of candidate governors for an incorrect PP-attachment in the sentence: "Jean voit la fille blonde de son balcon." ("Jean sees the blonde girl from his balcony."). The arc from 'fille' to 'de' is predicted by first-stage parsing, while the arcs from 'voit' to 'de' and from 'blonde' to 'de' represent alternative candidate governors for 'de'.

Learning a Correction Oracle

For the learning algorithm, the goal is to learn the oracle function $o : (G, d, C_d) \rightarrow (\hat{c}, l)$ that best predicts the governor $\hat{c}$ within the neighborhood of d with label l. As was the case for the transition oracle in transition-based parsing, the correction oracle serves as the engine for the corrector, navigating through possible dependency trees for a sentence by changing up to one arc at a time, starting from the originally predicted tree. The oracle should be optimized to be as accurate as possible: we want an o such that, for each pair (x, G_p) in an automatically parsed training set D_p, it will operate on the word forms of x to produce a tree that is as accurate as possible with respect to the gold tree G_g.

In order to learn an appropriate oracle, a ranking training set $D_p{}'$ must be derived from the automatically parsed training set D_p. To do this, we define the notion of a *gold oracle* for neighborhood correction, whose policy is to select the correct candidate if it is in C_d and otherwise select the original governor from first-stage parsing. This leads to a training set in which each tuple $(x, G_p) \in D_p$ is converted into a sequence of correction training tuples (G, d, C_d) with label (c, l) indicating the correct candidate in C_d and the correct functional role label l for the corresponding dependency arc. It should be noted that only correction steps where the correct governor is in C_d are included in the derived training set, as there needs to be a correct vector in order to create ranking constraints during model learning.

A ranking learner can then be viewed as training o to accurately predict, given features encoding information about the current graph, dependent, and set of candidate governors, the correct candidate to which the dependent should be attached.

Computational Complexity of Parse Correction

We now provide the space and time complexities for the parse correction algorithm. First of all, with respect to space complexity there isn't much to say: parse correction uses no additional data structures outside of the input parse tree, so the space complexity is clearly $O(n)$.

For time complexity, we now show that a corrector adhering to the neighborhood correction framework is linear in the length n of the input sentence, similar to the version of parse correction described by Hall and Novák (2005). First, we note that up to n dependents in a sentence are deterministically corrected in a single pass. We can see that for each such dependent d, the algorithm uses a linear model to select a new governor after extracting features for a local neighborhood set of candidate governors $C_d \subseteq N_m(d)$. If we can show that the size of $N_m(d)$ does not depend on n, then given that m is a low constant parameter value we have shown that the time complexity of neighborhood parse correction is $O(n)$.

To see why the size of $N_m(d)$ does not depend on n, we rely on the projectivity restriction that requires every $c \in C_d$ to be such that the dependency parse would remain projective if c were selected as the new governor of d. Since all candidates in $N_m(d)$ must pass through the original predicted governor c_o, we can divide $N_m(d) - \{c_o\}$ into subsets and bound their sizes separately:

- Candidates transitively governed by c_o. We begin with any candidates that are directly governed by c_o, essentially letting $m = 2$, and immediately we can note that at most two can lead to projective updates: the closest one linearly to the left c_l and to the right c_r of d, if they exist. If any other direct dependent w of c_o were chosen as the corrected governor of d, projectivity would be violated because w would not transitively govern every word form linearly between itself and d. If we continue incrementing m, we can see that at each step there are at most 2 new acceptable candidates. We have thus bounded the size of this set as being linearly dependent on m.

- Candidates transitively governed by the governor of c_o, not including the subtree headed by c_o. For this set, we can use the same argument as we did for those candidates transitively governed by c_o. There are still clearly at most two new valid candidates each time m is incremented, and these correspond to those that are closest to the left and to the right of d.

We can continue defining subsets of $N_m(d) - \{c_o\}$ as we have done above by moving up further to a new overarching governor, noting that at no point does n

need to be used to bound their size. With respect to m, we can see that, for $m \geq 1$, the size of the candidate set is bounded as follows:

$$|N_m(d)| \leq \sum_{k=1}^{m} 1 + 2(m-k). \tag{3.1}$$

While a tighter bound could potentially be achieved, this nonetheless means that the size of $N_m(d)$ is at most quadratic in m, giving us a final neighborhood correction time complexity of $O(m^2 n)$. □

3.1.3 Improving Context: Ranking and Features

Two crucial implementation aspects for the neighborhood correction framework that we did not discuss above are the learning algorithm used to train the linear model oracle, and the set of features to define over pairs of dependents and candidate governors with surrounding tree context. We will now describe the choices we made for these two implementation decisions, as well as how they differ from the work of Hall and Novák (2005) through the introduction of larger context.

Using a Ranking Learner

Our first large-context innovation is the use of a ranking learner for obtaining the optimal oracle. As described in Section 1.4, and similar to our implementation of the multiple candidate ARC-EAGER-MC transition based parsing approach described in Chapter 2, a linear model can be optimized directly to score the feature vector of the best candidate higher than the feature vectors of all other candidates.

The ranking approach differs from the binary classification learner used by Hall and Novák (2005). During learning, they essentially break down the prediction of a correct governor from among a set of k possible candidates into k different binary decisions, with a class of 1 assigned to the vector corresponding to the true governor and a class of 0 assigned to the vector of each other candidate. We believe that this learning approach matches less precisely the problem at hand, and ends up losing important context for each multiple-candidate decision during model optimization. For instance, imagine three arbitrary candidates a, b, and c. Suppose there exists an example in which a and b are candidates for dependent d, with a being correct, and there is another example in a different but similarly structured sentence (so G is the same with respect to the local surrounding context) in which b and c are candidates for d, with b being correct. A binary approach would have difficulty learning a model that requires the feature vector $\boldsymbol{v}_{b,d,G}$ over b, d, and G to simultaneously have class 0 and 1. On the other hand, a ranking approach easily deals with this situation

because it focuses on the differences between candidates: it requires only that the model $\boldsymbol{w}$ be such that $\boldsymbol{w}^T\boldsymbol{v}_{a,d,G} - \boldsymbol{w}^T\boldsymbol{v}_{b,d,G} > 0$ and $\boldsymbol{w}^T\boldsymbol{v}_{b,d,G} - \boldsymbol{w}^T\boldsymbol{v}_{c,d,G} > 0$, which is more feasible to model. We thus find that ranking is the most natural way to learn a linear model for a problem involving multiple choices represented using comparable feature vectors.

Features for Correction

Our second large-context innovation is our sensitivity to the need for sufficiently long-range features that will be useful for correcting parse errors. As we noted above, features for parse correction are defined over the candidate c, the dependent d, and the surrounding graph G.

In the work of Hall and Novák (2005), base features include a variety of information over three specific vertices: c, d, and the original predicted governor c_o from the parsing stage. The information over individual vertices includes word form, lemma, morphological tag, POS tag, and number of children, with an additional feature checking for case and number agreement between c and d. For higher-order features, they employ the commonly-used method in NLP of building combinations of features by hand, which we previously discussed in Section 2.3.1. Their feature combinations are fairly thorough, including combinations over two or three of the vertices using a range of information from each. However, we note that aside from the feature encoding number of children, the surrounding predicted tree context from the parsing stage is left entirely unused.

Keeping in mind that we have a special interest in difficult attachment types such as PP-attachment and coordination, it quickly becomes clear that a larger syntactic context for features is necessary if we hope to achieve meaningful results when correcting these attachments. The specific set of features we use for correction will be described later in Section 3.3.1 when presenting our experimental setup, but essentially we consider features that go a step further out along the predicted input parse tree. We also reiterate that our experiments make use of a polynomial kernel with degree $d = 3$ when learning SVM models, allowing for the implicit use of key feature combinations for PP-attachment and coordination. We will describe these combinations in more detail in our experimental setup.

3.2 Self-Trained Parsing with Correction

We have established an efficient two-stage dependency parsing system, which employs transition-based parsing in the first stage and neighborhood parse correction in the second stage. We now formalize a semi-supervised approach that could improve

two-stage dependency parsing, based on an existing approach that has proven useful for two-stage phrase-structure parsing. In Section 3.2.1, we present an overview of the generalized *self-training* framework for basic single-stage parsing, which is known to be largely ineffective outside of its application to domain adaptation. Then in Section 3.2.2 we describe the two-stage self-training framework, based on the seminal work of McClosky et al. (2006) that coupled a phrase-structure parser with a reranker, as well as a description of our particular implementation.

3.2.1 Basic Self-Training Framework

The idea of self-training is brought up often by researchers working on parsing, but it is traditionally viewed as being largely ineffective. Essentially, *self-training* is a method in machine learning that augments the training set of a model with new examples whose labels have been automatically predicted by the original model, and subsequently trains a new model from the augmented training set. The following describes the steps of basic self-training:

1. A model M is trained from the labeled training set D.
2. The model M is run on a new unlabeled set U.
3. A new model M' is retrained from a training set consisting of D and automatically labeled data from U.
4. The final output model is M'.

Intuitively, this approach does not seem capable of producing a model superior to the original, since the advantage of having more training data is counterbalanced by the fact that the quality of the new training data is only as good as that of the original model; it unlikely that the self-trained model will uncover any new insights. As discussed in the overview of self-training provided by McClosky et al. (2006), past attempts at using self-training for phrase-structure parsing have been largely futile, with for example the works of Charniak (1997) and of Steedman et al. (2003) including unsuccessful attempts at improving parsing through self-training.

A notable exception to this negative trend, however, involves research on *domain adaptation* for parsing. In this scenario, annotated data exists in a source domain but the goal is to learn a parser that works well over a target domain that has different lexical or syntactic characteristics. Many approaches for domain adaptation are rooted more in the machine learning aspects, with techniques including structural correspondence learning (Blitzer et al., 2006) to automatically induce correspondences among features from different domains, and a semi-supervised method

termed EasyAdapt++ (Daumé III et al., 2010) that augments the source domain feature space using features from unlabeled data in the target domain. But simple self-training has also been shown to be effective for domain adaptation, with improvements in phrase-structure parsing when performed over automatically parsed data from the target domain, for English (Sagae, 2010; Bacchiani et al., 2006) as well as for various other languages including French (Candito et al., 2011). We will include experiments regarding self-training for domain adaptation to medical text in this chapter.

3.2.2 Two-Stage Self-Training

Even when remaining within the same domain, however, *two-stage self-training* has recently proven to be effective in the work of McClosky et al. (2006), where a two-stage parser is used to annotate new data and only the first stage model is retrained. While their work is focused on phrase-structure parsing, with a PCFG parser in the first stage and a discriminative reranker in the second stage, the fundamental characteristics of the two-stage self-training framework for parsing can be described in a manner that is agnostic to the choice of syntactic representation and parsing approach. The following describes the steps of two-stage self-training:

1. A first-stage model M_1 is trained from the labeled training set D.
2. A second-stage model M_2 is trained based on the errors of M_1 on D.
3. The two-stage model (M_1, M_2) is run on a new unlabeled set U.
4. A new first-stage model $M_1{}'$ is retrained from a training set consisting of D and automatically labeled data from U.
5. The final output two-stage model is $(M_1{}', M_2)$.

Though not mentioned by (McClosky et al., 2006) this process can be considered to be a form of *up-training*, which is a scenario in which a training set is augmented with automatically labeled data that is of a higher quality than that which the original model could produce. Up-training has recently been used successfully for parsing English, with the setup involving a slow but highly accurate parser annotating new data, which is added to the original training set and used to improve the quality of a fast but less accurate parser (Petrov et al., 2010). Intuitively, then, we can see why self-training should be more effective for two-stage parsing than for single-stage parsing: the second reranking or correction stage improves the quality of the automatically labeled data, which may subsequently result

in a better self-trained first-stage parser. While intuition might also lead us to subsequently retrain the second-stage model, (McClosky et al., 2006) find for English that this leads to no additional improvement; we therefore follow their decision not to do this.

As mentioned earlier, our particular implementation of two-stage self-training involves dependency syntax, with the first stage corresponding to transition-based dependency parsing and the second stage corresponding to neighborhood parse correction. In our experiments, we test whether this implementation is as effective as that of the phrase-structure approach of (McClosky et al., 2006) with PCFG parsing and discriminative reranking. We do so both for in-domain French parsing and for domain adaptation to medical text, noting that McClosky and Charniak (2008) improved the performance of a reranking parser for English when adapted to biomedical text.

3.3 Correction Experiments

In this section we describe correction experiments for French, building on those for transition-based parsing presented in Chapter 2. Our experiments involve two-stage parsing, with a transition-based parser in the first stage and neighborhood parse correction in the second stage. We also test the effectiveness of the two-stage self-training approach of (McClosky et al., 2006) applied to dependency parsing instead of phrase-structure parsing, both in-domain and for domain adaptation.

3.3.1 Methods and Setup

We now present the setup of our correction experiments. For all methodological details concerning first-stage parsing, see our previous descriptions in Section 2.3. As was the case for those experiments, we used our own Python implementation of the algorithms and methods from this chapter; we reiterate that we plan to release a package of the code used in this thesis following its publication. The following are the novel elements of our methodology: creating the training, development, and test sections from the FTBDep for neighborhood correction; defining basic feature templates for correction; learning SVM correction models; processing external unlabeled corpora for self-training both in-domain and out-of-domain; and tuning of learning parameters.

FTBDep Correction Sets

When splitting the FTBDep into training, development, and test sets, we note that the development and test sets do not need to be explicitly created; rather, in order to evaluate neighborhood correction we simply run the trained two-stage parser on the FTBDep development and test sets, and then evaluate the final parse trees with respect to the parse trees obtained after only the first parsing stage.

For training, a jackknifing procedure must be used for parsing, similar to the one described in Section 2.3.1 for automatic POS tagging of the training set. In order to have an automatically parsed training set without using the evaluation sets, we did the following: the training set was split into ten parts, and then ten separate iterations were performed with a different held out set to be parsed with a transition-based parsing model trained on the other nine. We thus obtained an automatically parsed training set, from which we could then learn a neighborhood correction model. As mentioned earlier in Section 3.1.2, each sentence and corresponding pair of automatically parsed and gold projective dependency trees were then converted into a gold sequence of correction examples, with features for each example defined over the dependent, the set of candidate governors, and the current state of the corrected tree.

Feature Templates

Table 3.1 lists the basic feature templates for the oracle of neighborhood parse correction. The notation convention follows the one introduced in the experiments of Chapter 2, which includes shorthand terms for the type of information over one or more word forms, with subscripts indicating their locations in the current configuration. For instance, LEM_d indicates the lemma of the word form that is the current dependent whose governing arc is being corrected. The primary shorthand terms are LEM for lemma, POS for part-of-speech, and LAB for the functional role label in which the subscripted word form is the dependent. A new term is ISPR for whether a candidate is the originally predicted governor for a dependent in the graph output by a first-stage parser. The prefix H indicates information concerning the governor (if one currently exists) of a subscripted word form, and the prefix O indicates information concerning the dependent with arc label OBJ or DEP_COORD (if one currently exists) of a subscripted word form. The prefixes L and R have a different meaning than for transition-based parsing features, as they indicate information concerning the dependents of a subscripted c that are closest to a subscripted d on the left and on the right, respectively. Finally, NDEPS indicates the number of dependents, DIST indicates a binned linear distance between two word forms, DIR indicates the

Correction Templates	
Simple:	POS_d
	LEM_d
	OPOS_d
	POS_c
	LEM_c
	HPOS_c
	ISPR_c
Derived:	NDEPS_c
	$\text{DIST}_{c,d}$
	$\text{DIR}_{c,d}$
	$\text{PATH}_{c,d}$
	$\text{PUNC}_{c,d}$
	$\text{LPOS}_{c,d}$
	$\text{LLAB}_{c,d}$
	$\text{RPOS}_{c,d}$
	$\text{RLAB}_{c,d}$

Table 3.1: Basic feature templates for neighborhood parse correction, both simple ones over individual word forms and derived ones that use multiple words or surrounding syntactic structure.

direction of the linear distance between two word forms, PATH indicates the path along the graph between two word forms (represented as the number of upward steps followed by the number of downward steps from c to d), and PUNC indicates whether there is a punctuation mark linearly between two word forms.

The features for neighborhood correction that we use here are similar to those used for ARC-EAGER-MC transition-based parsing when comparing among multiple candidates with the RIGHT-ARC-MC(k, l) transition. A number of key new features are used here, however, that were not applicable for ARC-EAGER-MC: OPOS_d is possible because we have a full parse tree, while in ARC-EAGER-MC configurations never include right dependents of $\beta[0]$; ISPR_c and $\text{PATH}_{c,d}$ only make sense given a pre-existing predicted governor for d from the first-stage parser; $\text{DIR}_{c,d}$ is needed because candidates may be linearly on either side of d, while in ARC-EAGER-MC multiple candidates were considered only in the case of right-directed dependencies; and finally, the features with prefix L and R are potentially more useful than before by considering the most relevant contextual dependents of c with respect to d, instead of simply the left-most and right-most dependents of c independent of d.

As we noted earlier, the features presented here are more extensive than those used in transition-based parsing (cf. Chapter 2) or in related work on parse correction (Hall and Novák, 2005), and they are designed to take into account important information for difficult attachments, with the OPOS_d feature being key. For PP-

attachment, the distribution of a prepositional phrase differs greatly depending on whether its object is a verb or a noun. Similarly, for a coordinating conjunction d, syntactic requirements include the matching of POS categories for its two conjuncts, which are the governor of d and the DEP_COORD dependent of d.

SVM Learning

As previously explained in Section 1.4, there are two main machine learning approaches we use to optimize the oracle correction function: multiclass SVM and ranking SVM, each using a polynomial kernel of degree d=3. As in our previous experiments in Chapter 2, we note that SVM model training and prediction was performed using the LIBSVM package (Chang and Lin, 2011) for multiclass models, and the SVM^{light} package (Joachims, 1999) for ranking models.

For the neighborhood oracle we use both multiclass and ranking SVM approaches to learn two models that jointly represent the oracle. Given a set of k candidate governors for a dependent, a separate feature vector is constructed for each candidate and a ranking model is used to select the best one. If the new governor is different from the one predicted by the first-stage parser, then a label l for the new arc is chosen by a separate labeling classifier using the feature vector of the new governor.

As we noted earlier, the use of a cubic kernel allows the ranking model to implicitly use feature combinations useful for PP-attachment and coordination. Specifically, these are the feature combinations over the lemmas or POS categories of the three vertices c, d, and d_o. For PP-attachment, subcategorization or selectional preference is implicitly encoded with feature triples over the POS category of c, the lemma of d, and the POS category of d_o. And for coordination, the test of conjunct POS matching are implicitly encoded with feature triples over the POS categories of c and d_o, with the POS category of d needed as well in order to separate out cases of coordination (where the POS category is CC) from other possibilities.

External Corpora for Self-Training

For experiments concerning two-stage self-training for parsing, which we presented in Section 3.2, we now describe our choices of external unannotated corpora to test, as well as the preprocessing carried out.

For our standard in-domain experiments, we tested two unannotated corpora of French news that remain within the journalistic domain of the FTBDep, which we recall is composed of French news articles from the *Le Monde* newspaper. The first corpus, which is freely available for research purposes, consists of articles from

the *L'Est Républicain (ER)* regional newspaper[1] and contains around 9 million sentences. The second corpus we use, which is unfortunately not freely available, consists of dispatches from the *Agence France Presse (AFP)* and contains around 4 million sentences.

For our domain adaptation experiments, we used a French text corpus from the medical domain to contrast with the journalistic domain of the FTBDep. This corpus was first used in a previous work on domain adaptation for phrase-structure French parsing (Candito et al., 2011), and consists of texts from the European Medicines Agency, specifically the French part of the EMEA section of the OPUS corpus (Tiedemann, 2009). It consists of documents summarizing European public assessment reports on specific medicines. After converting the original PDF files to text and doing some preprocessing, Candito et al. (2011) built small evaluation sets (574 development and 544 test sentences) from the resulting 267,000 sentences. They manually annotated these sets for phrase structure, and later automatically converted them to dependencies (Candito and Seddah, 2012a) using the same procedure as was used for converting the FTB to the FTBDep. For our domain adaptation self-training experiments, we used the unannotated portion of the EMEA for automatic parsing, with the EMEA development and test sets used as reference sets for evaluation.

Each unannotated corpus was preprocessed using the Bonsai tool[2], which performed sentence segmentation, word tokenization, and additionally, consistent with our automatic annotation of the FTBDep in Section 2.3, performed automatic POS tagging with the MElt package and lemmatization with the Lefff lexicon.

Tuning Learning Parameters

Before performing the final evaluation of two-stage parsing on the FTBDep test set, we tuned some of the LIBSVM and SVM^{light} learning parameters using the FTBDep development set. As was the case in our tuning for transition-based parsing in Chapter 2, the parameter values leading to the highest LAS and UAS over sentences in the development set were identified here, and subsequently locked in for the final evaluation over the test set.

The learning parameters we fixed from the start were the same as those for transition-based parsing: the use of the C-SVC learning type for SVM, as described in Section 1.4.2, as well as the use of a polynomial kernel with d=3. As for the other parameters, tuning actually led us to the same parameters as those for the transition-based parsing experiments. We settled on an inner coefficient of s=0.1 for

[1] http://www.cnrtl.fr/corpus/estrepublicain/

[2] http://alpage.inria.fr/statgram/frdep/fr_stat_dep_parsing.html

Model Group	Fine POS Categories
Adjectives	ADJ
Adverbs	ADV
Conjunctions	CC, CS
Nouns	NC, NPP
Prepositions	P,P+D
Pronouns	PRO, PROREL
Verbs	V, VINF, VPP, VPR, VS

Table 3.2: Grouping of classification and ranking models by fine POS category of dependent word forms for neighborhood parse correction.

the polynomial kernel (cf. Equation 1.6), a C-SVC regularization parameter with value C=1 for multiclass models and C=0.05 for ranking models, and a C-SVC termination criterion parameter with value ϵ=1.0 for both multiclass and ranking models. Finally, we decided to use no class bias for any of our models.

A new parameter introduced for neighborhood correction is the size m of the neighborhood from which to select candidate governors. We used a gold oracle corrector over the FTBDep development set to find a good value for m and found that as m increases, the average number of candidate governors per dependent increases approximately linearly, while the improvement in topline accuracy tapers off rather quickly. We ended up settling on m=3, as it afforded a large increase in topline accuracy over m=2, while m=4 did not seem to improve the topline accuracy enough over m=3 to justify the increase in average number of candidate governors per dependent. Topline accuracy for gold oracle correction will be discussed shortly when we present the results for our experiments.

As we did for transition-based parsing, a last modification we made during development was to split each model further depending on the POS category of the word form at the front of the buffer in a configuration. The POS groupings we used are listed in Table 3.2. As we noted before, this was mainly done for reasons of computational efficiency, as it significantly reduces the overall time and space required for training and parsing. Additionally reflected in the table is the fact that we ignored certain dependents for correction. During development, we found that dependents with certain POS categories are unnecessary to treat, for a number of reasons: rarity (wh-words, interjections, prefixes, imperative verbs, and prepositions coupled with pronouns), accurate handling in first-stage parsing (clitics and determiners), difficulty in modeling (foreign words), and non-scoring tokens (punctuation).

Parsing System	Overall		Preps	Coords
	LAS	UAS	UAS	UAS
ARC-STANDARD	87.3	89.9	83.8	59.2
+GOLD CORRECTION	94.3	94.8	93.2	80.5
+MODEL CORRECTION	87.7*	90.3*	84.3*	63.7*
ARC-EAGER	87.1	89.7	84.0	64.4
+GOLD CORRECTION	93.9	94.4	93.1	82.8
+MODEL CORRECTION	87.5*	90.2*	84.6*	67.4*

Table 3.3: LAS and UAS results, in percent, over the FTBDep test set when using either a gold correction oracle or an automatically trained model correction oracle over first-stage trees output by ARC-STANDARD and ARC-EAGER baseline transition-based parsers. Also includes UAS results when restricting scoring dependents to prepositions (P,P+D) or coordinating conjunctions (CC). $*$ indicates a significant improvement over the baseline.

3.3.2 Results

This section describes the results of our two-stage parsing experiments, with first-stage transition-based parsing and second-stage neighborhood correction, as well as our experiments that use self-training with a large external text corpus to try to improve both in-domain and out-of-domain parsing.

Neighborhood Correction

We present our results for neighborhood correction using two types of oracles: (i) gold correction oracles that always choose the correct governor (if it appears in the candidate set) for a dependent and give us a topline accuracy for neighborhood correction; (ii) the actual trained oracle models. Table 3.3 shows the evaluation results on the FTBDep test set when using either a gold oracle or an automatically trained model oracle in the second-stage corrector, with either ARC-STANDARD and ARC-EAGER first-stage transition-based parsing. The table includes LAS and UAS results over all word forms, as well as UAS results when considering difficult attachments (those with either preposition or coordinating conjunction dependents). As previously noted in our experiments in Chapter 2, the scoring word forms in the test set amount to 31,404 total, with 5,706 prepositions and 801 coordinating conjunctions. Punctuation marks are again not scored.

As we can see from the gold oracle results, the top line accuracy corresponds to an error correction rate of around 50% across the two transition-based parsers and the POS categories of dependent tokens to evaluate. While the top line may seem low, parse errors from the first stage mostly correspond to difficult, rare, or ambiguous parts of a sentence's syntactic structure; a 50% correction rate thus seems like an adequate top line. It is important to acknowledge, though, that parse

correction with a restricted neighborhood size (m=3 in our experiments) cannot be expected to correct most errors.

Moving on to actual accuracy improvements achieved by our trained oracle models, increments in accuracy for neighborhood correction are modest but statistically significant in all cases. When scoring all tokens, we see +0.4-0.5 improvements in LAS and UAS for both transition-based parsers. When scoring preposition tokens, we see a +0.5 UAS improvement for the ARC-STANDARD parser and a +0.6 UAS improvement for the ARC-EAGER parser. Finally, when scoring coordinating conjunction tokens, we see a +4.5 UAS improvement for the ARC-STANDARD parser and a +3.0 UAS improvement for the ARC-EAGER parser. We find these results to be encouraging, and a validation of the neighborhood correction approach for two-stage parsing. Having access to rich surrounding syntactic context from the first-stage parser's output appears to be very useful when reconsidering certain difficult attachments, especially coordination.

Comparing our results to those found in related works for Czech and English, our improvements do not immediately appear to be as high. Hall and Novák (2005) apply parse correction to obtain a +1.2 improvement in UAS (over an 81.6 baseline) when using the Collins phrase-structure parser (Collins et al., 1999) applied to Czech dependency parsing, and a +0.7 improvement in UAS (over an 84.4 baseline) when using the Charniak phrase-structure parser (Charniak, 2001) adapted to Czech dependency parsing (Nivre and Nilsson, 2005). Note that the Charniak parser, which has the higher baseline of the two, achieves improvements comparable to ours. For English, Attardi and Ciaramita (2007) apply tree revision learning to obtain large improvements when using two versions of the shift-reduce DeSR parser (Attardi, 2006): for the first version they obtain a +5.3 improvement in UAS (over an 85.0 baseline), while for the second version they obtain a +0.7 improvement in UAS (over an 88.4 baseline). In both cases, though, they do not actually improve over the UAS (90.3) of a prominent shift-reduce parser for English (Yamada and Matsumoto, 2003).

In the related work, we thus see a trend of lower baselines leading to higher correction rates. This is understandable, as one might expect the additional errors produced by a low-baseline parser to be easier to correct. We would argue that the correction rate over a low-baseline parser is not particularly informative, however; we are interested in seeing whether a corrective approach can improve on a parser that has accuracy competitive with the state-of-the-art for a language. The +0.7 improvements in UAS in the related works when using the higher baseline parsers are thus what we compare against our results, which end up having a similar +0.6 improvement in UAS over ARC-STANDARD transition-based parsing.

All	FS+	FS–
SS+	28,117	204
SS–	64	3,019

P,P+D	FS+	FS–
SS+	4,778	47
SS–	17	864

CC	FS+	FS–
SS+	505	35
SS–	11	250

Table 3.4: Confusion matrices for first-stage (FS) ARC-EAGER parsing and for second-stage (SS) neighborhood correction, with each scored word form attached correctly (+) or incorrectly (–) after each stage. Three scoring settings are considered: all word forms (All), prepositions (P,P+D), and coordinating conjunctions (CC).

A final result concerning neighborhood correction is the extent to which this second stage modifies the original parse tree. Table 3.4 shows confusion matrices for attachment results when using ARC-EAGER transition based parsing, for the three UAS scoring settings (all word forms, only prepositions, and only coordinating conjunctions). We can see that the number of errors introduced tends to be low compared to the number of errors corrected. For prepositions, for instance, there were only 17 errors introduced compared to 47 errors corrected.

In-Domain Self-Trained Two-Stage Parsing

We now turn to results for self-training to improve two-stage parsing within the journalistic domain. To reduce the number of parameters, we restricted this experiment to a two-stage parsing system with an ARC-EAGER parser in the first stage. To re-iterate, this experiment involved the following steps: (i) the automatic parsing of an external corpus using a two-stage parsing system; (ii) the retraining of the first-stage transition-based ARC-EAGER parser using a combination of gold FTBDep and automatically parsed external sentences; (iii) the evaluation of the new two-stage parsing system on the FTBDep development and test sets.

Because crucial factors for this evaluation are the choice of external corpus to parse, the amount of external data to use, and the ratio of FTBDep to external data in the combined training set, we have decided to include here our results for the tuning portion of these parameters on the FTBDep development set.

Table 3.5 shows overall LAS and UAS for two-stage self-training on the FTB-Dep development set. The parameters evaluated were: (i) whether to use the EP or the AFP external corpus; (ii) whether to do only first-stage parsing of the external

External	Original	LAS	UAS
Baseline (None)	FTBDep (10k)	86.4	89.1
AFP one-stage (10k)	FTBDep (10k)	86.3	89.0
AFP one-stage (10k)	FTBDep (50k)	86.2	88.9
AFP one-stage (50k)	FTBDep (10k)	86.4	89.1
AFP one-stage (100k)	FTBDep (10k)	86.3	89.0
AFP two-stage (10k)	FTBDep (10k)	86.5	89.2
AFP two-stage (10k)	FTBDep (50k)	86.4	89.0
AFP two-stage (50k)	FTBDep (10k)	86.6†	89.3†
AFP two-stage (100k)	FTBDep (10k)	86.4	89.1
ER one-stage (10k)	FTBDep (10k)	86.1	88.9
ER one-stage (10k)	FTBDep (50k)	86.0	88.8
ER one-stage (50k)	FTBDep (10k)	86.2	88.9
ER one-stage (100k)	FTBDep (10k)	86.3	89.0
ER two-stage (10k)	FTBDep (10k)	86.2	89.0
ER two-stage (10k)	FTBDep (50k)	86.1	88.8
ER two-stage (50k)	FTBDep (10k)	86.3	89.0
ER two-stage (100k)	FTBDep (10k)	86.3	89.1

Table 3.5: First-stage ARC-EAGER parsing LAS and UAS results, in percent, over the FTB-Dep development set, listed according to the choice of external corpus (AFP or ER), the method for parsing the external corpus (one-stage or two-stage), and the number of sentences (in thousands) from each corpus in the final training set for ARC-EAGER. † indicates the best result, though not statistically significant over the baseline.

corpus or two-stage parsing and correction; (iii) how many external sentences to add to the new training set; and (iv) how much to weight the original FTBDep sentences in the new training set. Note that we use a simple method for weighting up original FTBDep sentences: to weight up by a factor of five, we simply add four additional copies of each original FTBDep sentence to the new training set. We use the notation 10k to approximate the 9,881 original FTBDep training sentences, and 50k to denote the number of sentences corresponding to a weighting up by a factor of five.

These initial development set results were not great, as the majority of the self-training settings lead to LAS and UAS that are not any higher — and in many cases lower — than that of the baseline two-stage parsing system. As expected, using automatically parsed sentences from a two-stage system (parsing and correction) leads to better results across the board compared to using automatically parsed sentences from a one-stage system (parsing only). Interestingly, using the AFP external corpus leads to better results than using the ER corpus, which may be an indication of its higher similarity and relevance to the FTBDep corpus. This may be due to the fact that the AFP corpus consists of newswires from an agency that covers

Parsing System	Overall		Preps	Coords
	LAS	UAS	UAS	UAS
Baseline two-stage	87.5	90.2	84.6	67.4
Self-trained two-stage	87.3	90.1	84.5	64.5

Table 3.6: LAS and UAS results, in percent, over the FTBDep test set when using a two-stage parsing system of ARC-EAGER transition-based parsing followed by correction. The baseline setting uses this system alone, while the self-trained setting uses a two-stage self-trained parser. Also includes UAS results when restricting scoring dependents to prepositions (P,P+D) or coordinating conjunctions (CC).

major French national and international news, which is closer to the internationally recognized *Le Monde* newspaper whose articles make up the FTBDep corpus; in contrast, the ER corpus consists of articles from a regional French newspaper that covers different types of news and is perhaps written in a different style.

With respect to the amount of external data used and the ratio of original to external data, the best setting combines 50k external sentences with the original 10k FTBDep sentences (no weighting up of the original sentences). Although the study of McClosky et al. (2006) tries using many more external sentences, up to two million, this is not practical in terms of both space and time given our use of SVM learning and our available amounts of computer processing power and memory; it is possible that the PCFG learning and parsing algorithms used in their study scale better to larger training sets. We also note, however, that our self-training results seem to degrade when going up from 50k to 100k external sentences, so we are in any case not optimistic about a further improvement when using additional external data.

Given the above development results, for the final test set evaluation we used parameter settings corresponding to the highest increase in LAS and UAS over the two-stage parsing baseline: 50k external sentences taken from the AFP corpus, automatically parsed using a two-stage parser, with no additional weighting of the original 10k FTBDep training sentences. Table 3.6 shows overall LAS and UAS for this final evaluation of two-stage self-training on the FTBDep test set. We can immediately see that the results are disappointing: by all four LAS and UAS measures, two-stage self-training does not improve upon the simpler two-stage parsing system that we presented earlier. In fact, results appear to degrade slightly for some of the measures, most notably for coordinating conjunction UAS.

Trying to understand the lack of satisfying self-training results for our two-stage dependency parsing system, we consider a number of explanations. With respect to the SVM learning process for transition-based parsing, it is possible that it doesn't handle noisy input data well. A transition-based parser needs to make

accurate sequential decisions on transitions between configurations, with each mistake further derailing the course of parsing. Perhaps PCFG phrase-structure parsing systems are more robust to noisy training data, due to their use of probability estimations for derivation rules. A second consideration, which we believe to be very important, is that in the work of McClosky et al. (2006) for phrase-structure parsing there is a much larger difference in initial performance between the first-stage parser and the two-stage system (89.7 and 91.3 f-score, respectively) than we obtained for dependency parsing with our two-stage system (89.7 and 90.2 UAS, respectively). As we have previously mentioned, empirical results from experiments in self-training and up-training indicate that self-training does not work unless the external data is parsed more accurately than it would be by the first-stage parser alone. It stands to reason that the higher the difference in accuracy, the more likely we are to see an improvement in the self-trained first-stage parser; perhaps a 0.5 UAS difference in accuracy between one- and two-stage parsing is not large enough to outweigh the drawback of introducing noisy examples into the parser's training set.

Self-Training for Domain Adaptation

We now turn to results for self-training to improve domain adaptation in transition-based parsing, with a source journalistic domain and a target medical domain. To reduce the number of parameters, we again restricted the experiment to a two-stage parsing system with an ARC-EAGER parser in the first stage. This experiment involved slightly modified steps compared to those for in-domain self-training: (i) the automatic parsing of an unlabeled external medical corpus (EMEA) using a two-stage parsing system; (ii) the retraining of the first-stage transition-based ARC-EAGER parser using a combination of FTBDep journalistic domain and automatically-parsed EMEA medical domain sentences; (iii) the evaluation of the new two-stage parsing system on both the FTBDep and EMEA evaluation sets.

As we did before for the in-domain self training experiment, we include here our parameter tuning results on the EMEA development set with respect to the amount of external data to use and the ratio of FTBDep to external data in the combined training set. Table 3.7 shows overall LAS and UAS for two-stage self-training on the EMEA development set. We again use a simple method for weighting up original FTBDep sentences: to weight up by a factor of five, we add four additional copies of each original FTBDep sentence to the new training set. We use the notation 10k to approximate the 9,881 original FTBDep training sentences, and 50k to denote the number of sentences corresponding to a weighting up by a factor of five.

These initial development set results were fairly encouraging. As expected, using automatically parsed medical text to help retrain a first-stage ARC-EAGER

External	Original	LAS	UAS
Baseline (None)	FTBDep (10k)	84.2	87.0
EMEA two-stage (10k)	FTBDep (10k)	84.6	87.4
EMEA two-stage (10k)	FTBDep (50k)	84.7	87.6
EMEA two-stage (50k)	FTBDep (10k)	84.4	87.2
EMEA two-stage (100k)	FTBDep (10k)	84.3	87.1

Table 3.7: First-stage ARC-EAGER parsing LAS and UAS results, in percent, over the EMEA development set, listed according to the number of sentences (thousands) for each corpus in the final training set for ARC-EAGER.

	FTBDep Test		EMEA Test	
Parsing System	LAS	UAS	LAS	UAS
Baseline two-stage	87.5	90.2	86.0	88.3
Self-trained two-stage	87.4	90.0	86.2	88.3

Table 3.8: LAS and UAS results, in percent, over the journalistic FTBDep and medical EMEA test sets when using a two-stage parsing system of ARC-EAGER transition-based parsing followed by correction. The baseline setting uses this system alone, while the self-trained setting uses a two-stage self-trained parser adapted to the medical domain.

parser resulted in better first-stage parsing performance over medical sentences. This was true across different settings for the amount of external data to use and the ratio of original to external data, though the best setting combined 10k external sentences with the original FTBDep sentences weighted up to 50k. This is notably different from our in-domain self-training experiments, in which the best setting combined 50k external sentences with the 10k original FTBDep sentences.

Given the above development results, for the final test set evaluation we used parameter settings corresponding to the highest increase in LAS and UAS over the two-stage parsing baseline: 10k external sentences taken from the EMEA corpus, automatically parsed using a two-stage parser, with a weighting up of the original FTBDep training sentences to 50k. Table 3.8 shows overall LAS and UAS for this final evaluation of two-stage self-training over the FTBDep and EMEA test sets, this time adding neighborhood correction on top of the self-trained first-stage ARC-EAGER parser.

The results are again, as for our in-domain self-training experiments, disappointing. The self-training two-stage parser adapted to the medical domain performed only slightly better than the baseline two-stage parser on the EMEA test set (+0.2 LAS), while the performance on the FTBDep test set decreased slightly. It is interesting that the somewhat more substantial increase over the baseline we observed over the EMEA development set did not translate to the EMEA test set. This is perhaps a consequence of using small evaluation sets for EMEA (574 de-

velopment and 544 test sentences) compared to the more robust evaluation sets for the FTBDep (1,235 development and test sentences each). Recalling our discussion of in-domain self-training experiments, other explanations for why the self-training approach to domain adaption did not work well in our experiments include the possibility that the automatically parsed sentences from the unannotated portion of the EMEA were too inaccurate, or that the SVM learning process for transition-based parsing might just not handle noisy input well.

Chapter 4

Parsing with Generalized Lexical Classes

Toutes les généralisations sont fausses, y compris celle-ci.

— *Alexandre Dumas*

In the preceding chapters we explored the first main thread of our thesis: the search for a framework where more syntactic context is available for attachment decisions in dependency parsing while retaining computational efficiency. In Chapter 2 our approach was to incorporate directly into a transition-based parser a way to consider multiple candidate governors simultaneously for a dependent. Then in Chapter 3 we considered a two-stage parsing approach where a parse correction model reconsiders attachments made by the parser, with the dual benefits of richer syntactic context as well as the ability to efficiently compare multiple candidate governors simultaneous for a dependent. We now turn to the second main thread of our thesis: the creation and use of automatically-built lexical resources for semi-supervised dependency parsing, with the parsing and correction algorithms from the previous chapters serving as a backbone. In this chapter, we specifically investigate different types of lexical classes that can be used in place of word forms for features during parsing and correction. Our discussion primarily follows a previously published work of ours (Henestroza Anguiano and Candito, 2012); key differences include our use here of SVM learning, as well as our decision to not do probabilistic weighting of lexical classes in parsing features.

A central problem in data-driven approaches for dependency parsing is the accurate modeling of lexical relationships from potentially sparse counts in a training corpus. Because treebanks used for training are often small, lexical features may appear infrequently during training, especially for languages with richer morphology than English. This may impede a parser's ability to generalize outside of its training set with respect to lexical features. Our work thus focuses on reducing lexical data sparseness through the use of generalized lexical classes. In Section 2.1.2 we already saw the benefits of generalizing word forms with lemmas, and in this chapter we additionally investigate distributional word clusters and semantic synonym sets as useful classes. In applying lexical classes to parsing, a novel aspect of our approach compared to related work is that we allow for the assignation of multiple classes of the same type to a word form: for instance, we allow for a word form to belong to multiple synonym sets, which is subsequently reflected in feature vectors by having multiple indicator features fire for a single categorical feature template (cf. Section 1.4.3).

In Section 4.1, we describe the methods in distributional lexical semantics needed to create from a large text corpus a distributional thesaurus, which is used as the basis for the construction of our generalized lexical classes. We provide an overview of the methods used by NLP researchers in the past, settling on the intuitive and concise formal representations used by Lin (1998) and Curran (2004); this leads to a description of different metrics for weighting and similarity we consider

for building a distributional thesaurus, with an explanation for why we chose these metrics over others. We then present generalized lexical classes in three spaces we have chosen to investigate, each relying on a distributional thesaurus: (i) lemmas, with multiple assignation for word forms obtained using other lemmas with high similarity in the thesaurus; (ii) clusters, generated using the thesaurus and with single assignation for word forms; and (iii) synsets, obtained using WordNet coupled with the thesaurus through automatic sense ranking, which enables multiple assignation for word forms.

In Section 4.2 we then present experiments for French where we test the use in parsing models of lexical features derived from our three spaces of lexical classes, with either single or multiple assignation. The parsing approaches used as the basis for our experiments are those we previously investigated in Chapters 2 and 3.

4.1 Distributional Lexical Semantics and Classes

The distributional hypothesis states that words occurring in the same contexts tend to have similar meanings, as posited by Harris (1954) in an often cited work. We focus on methods that define a measure of *distributional similarity* between lexical terms based on the similarities between their contexts, and can thus generate a distributional thesaurus from a large collection of lexical terms and contexts. An entry in a distributional thesaurus contains, for each lexical term, a list of distributionally similar terms ordered by similarity. Example entries are shown in Table 4.1.

It is important to note that high distributional similarity does not necessarily indicate *synonymy*, which is a relation in which two words have the same meaning, but rather a high level of semantic similarity, which can take other forms. For example, the relation of *antonymy*, which indicates an opposite meaning, can be found between two distributionally similar adjectives 'tragique' ('tragic') and 'comique' ('comic'). And in Table 4.1 the relation of *hyponymy*, which indicates that one is an instance of another, is observed with 'véhicule' ('vehicle') being found similar to 'voiture' ('car'). However, this does not pose a problem for us because we are interested in the replaceability of word forms in syntactic structure, abstracting away from the precise nature of the semantic relationships between distributionally similar word forms. Even though 'véhicule' and 'voiture' are not synonyms, the fact that they have similar syntactic distributions means that it might be advantageous to group them into a single generalized lexical class for the purpose of alleviating data sparseness in syntactic parsing.

We first present in Section 4.1.1 an overview of related work on distributional lexical semantics, whose methods can be used to automatically generate thesauri,

véhicule	voiture−0.490, camion−0.387, engin−0.307, avion−0.295, bus−0.256, navire−0.253, camionnette−0.253, train−0.250, automobile−0.248, appareil−0.244, tracteur−0.241, fourgon−0.241, bateau−0.238, moto−0.237, matériel−0.230, hélicoptère−0.218, machine−0.214, autobus−0.214, ...
calciner	carboniser−0.200, brûler−0.122, consumer−0.119, déchiqueter−0.110, éventrer−0.109, endommager−0.108, gésir−0.100, incendier−0.099, broyer−0.097, momifier−0.095, cribler−0.094, recouvrir−0.092, joncher−0.092, noircir−0.092, abîmer−0.091, empiler−0.091, ensevelir−0.089, ...

Table 4.1: Lemmatized distributional thesaurus entries in French for the noun 'véhicule' ('vehicle') and verb 'calciner' ("to calcify"). Neighboring lemmas are listed by descending similarity.

and also describe related work on using generalized lexical classes to improve parsing. We subsequently present in Section 4.1.2 the specific framework we use to automatically generate a thesaurus through distributional lexical semantics, including a discussion of the weighting and similarity metrics that we have chosen to test. Finally, in Section 4.1.3 we define the generalized lexical class spaces of lemmas, clusters, and semantic senses that we work with in our experiments.

4.1.1 Related Work

Distributional lexical semantics is an area that has been studied extensively in NLP, with the main thread of research focusing on metrics of similarity derived from statistical frequency analyses of terms (words) and contexts (n-grams, text windows or grammatical dependencies, etc.) in large text corpora. Applications for distributional lexical semantics are numerous, and, as noted by Turney and Pantel (2010) in a survey of vector space approaches to distributional similarity, these include word clustering, word classification, automatic thesaurus generation, word sense disambiguation, context-sensitive spelling correction, semantic role labeling, query expansion, textual advertising, and information extraction, to name a few. The application that we are interested in is the creation of distributional lexical classes for parsing, for which there has also been a substantial amount of research conducted in NLP.

We first present related work on methods for distributional lexical semantics in general, including ways to evaluate distributional metrics and thesauri. We subsequently look at related work in which generalized lexical classes are used to improve parsing performance.

Methods in Distributional Lexical Semantics

Research on distributional lexical semantics goes as far back as the work of Miller and Charles (1991). Coming from a psycholinguistic background, the authors investigated the relationship between semantic and contextual similarity for pairs of nouns varying from high to low semantic similarity, with semantic similarity estimated by subjective ratings and contextual similarity estimated by the method of sorting sentential contexts. Their results indicated that the more often two words can be substituted into the same contexts, the more similar in meaning they are judged to be.

In the field of NLP, an early study on distributional lexical semantics was that of Pereira et al. (1993). In that work, nouns and verbs were clustered according to their distribution in particular syntactic contexts; specifically, using pattern matching to find pairs containing a verb and a noun dependent in a large text corpus. Empirical distributions of a noun with respect to verbs (or vice versa) were used to calculate distributional similarity between like objects, with the metric of choice being relative entropy, or Kullback-Leibler (KL) distance, between two distributions. After clustering words using a number of different approaches, the authors found that resulting clusters were intuitively informative and led to class-based word cooccurrence models with substantial predictive power. Another important early work is that of Dagan et al. (1994, 1997), who automatically extracted similar words in order to help alleviate the problem of data sparseness in statistical natural language processing. They found better results for word sense disambiguation when using similarity-based smoothing compared to standard back-off smoothing methods. Additional early studies on distributional lexical semantics have also paved the way for research in this area (Hindle, 1990; Grefenstette, 1994), with methods for analyzing word collocations comprising a chapter of a widely-used NLP textbook (Manning and Schütze, 1999).

The works we regard to be the most important with respect to our own are those of Lin (1998) and of Curran (2004). These works specifically concern the automatic construction and evaluation of distributional thesauri, and helpfully decompose the problem of distributional similarity calculation from corpus statistics down into separate steps that can each be handled in different ways. The work of Lin (1998) has the contribution of introducing an objective evaluation method for automatically constructed thesauri, which compares them to manually constructed thesauri or related resources such as WordNet (Fellbaum, 1998). The subsequent work of Curran (2004) contains a large survey of different methods for calculating distributional similarity, and we borrow its elegant terminology that distills the process of constructing a distributional thesaurus down to its basic components.

The work of Lin (1998) is particularly important because it was the first, to our knowledge, that set up a well-defined evaluation of distributional thesauri against a similar gold reference resource. Prior to that work, as Lin (1998) notes, methods for evaluating automatically constructed thesauri relied on indirect tasks or subjective judgments. Such approaches include that of Smadja (1993), in which automatically extracted collocations were judged manually by a lexicographer, and those of Dagan et al. (1993) and of Pereira et al. (1993), in which clusters of similar words were evaluated by how well they were able to recover data items removed from the input corpus one by one. After Lin (1998), other works have used his general approach in evaluating distributional methods with manually built thesauri (Curran, 2004; Van Der Plas and Bouma, 2004), with some works introducing new metrics for evaluation, such as a TOEFL test (Ferret, 2010) or entity sets gathered from Wikipedia (Pantel et al., 2009).

As pertains to French, previous work on the creation and evaluation of distributional thesauri exists but is not as extensive. Distributional methods have certainly been used in French, notably in the work of Bourigault (2002) on Upery, a distributional analysis module that calculates proximities between words and their contexts, and the work of Ferret (2004), which used distributional similarity to build word senses from a network of lexical co-occurrences. The differences between their work and ours lies primarily in the area of application: for Bourigault (2002) it was the construction of ontologies, and for Ferret (2004) the automatic discovery of word senses. To our knowledge, we are the first to perform an evaluation of distributional thesauri in the style of Lin (1998) for French, with the evaluation presented in this chapter taken from a previously published work (Henestroza Anguiano and Denis, 2011).

It should also be noted that recent research in distributional lexical semantics has begun to include more sophisticated approaches, including Bayesian estimation of distributional similarities (Kazama et al., 2010) and latent variable models (Chrupala, 2011). Though we have not used these approaches, they are promising avenues for future research on using distributional lexical semantics to improve parsing.

Generalized Lexical Classes for Parsing

The use of word classes for parsing dates back to early research on generative phrase-structure parsing, whether using semantic classes obtained from hand-built resources or less-informed classes created automatically. Bikel (2000) tried incorporating WordNet-based word sense disambiguation into a parser, but failed to obtain an improvement. Xiong et al. (2005) generalized bilexical dependencies in a generative parsing model using the Chinese semantic resources CiLin and HowNet, obtaining

improvements for Chinese parsing. More recently, Agirre et al. (2008) showed that replacing words with WordNet semantic classes could slightly improve English generative parsing. Lin et al. (2009) used the HowNet resource within the split-merge PCFG framework (Petrov et al., 2006) for Chinese parsing: they used the first-sense heuristic to append the most general hypernym to the POS of a token, obtaining a semantically-informed symbol refinement, and then guided further symbol splits using the HowNet hierarchy. For French, recent approaches using a PCFG-LA framework have improved parsing accuracy with lexical classes. Sigogne et al. (2011) used the Lexicon-Grammar resource (Gross, 1994) to cluster verbs based on shared syntactic characteristics (e.g. transitivity), and a follow-up work used the Lefff lexicon to cluster verbs based on shared subcategorization frames (Sigogne and Constant, 2012). Candito and Crabbé (2009), on the other hand, obtained improvements in French parsing using less-informed unsupervised word clusters from a large unannotated text corpus calculated with the Brown algorithm (Brown et al., 1992).

In dependency parsing, word classes are integrated as features in underlying classification models. In a seminal work, Koo et al. (2008) used Brown clusters as features in a graph-based parser, improving parsing for both English and Czech. However, attempts to use this technique for French have lead to no improvement when compared to the use of lemmatization and morphological analysis (Candito et al., 2010b). Sagae and Gordon (2009) augmented a transition-based English parser with clusters using unlexicalized syntactic distributional similarity: each word was represented as a vector of counts of emanating unlexicalized syntactic paths, with counts taken from a corpus of auto-parsed phrase-structure trees, and HAC clustering was performed using cosine similarity. For semantic word classes, Agirre et al. (2011) integrated WordNet senses into a transition-based parser for English, reporting small but significant improvements in LAS (+0.26 with synsets and +0.36 with semantic files) on the full Penn Treebank with first-sense information from Semcor.

Concerning techniques for out-of-domain parsing, we recently participated in work that used word clustering for domain adaptation of a phrase-structure PCFG-LA parser for French, deriving clusters from a "bridge" corpus containing text from both source and target domains, with parsing improvements observed in both domains (Candito et al., 2011). Other than the recently published work on which this chapter is based (Henestroza Anguiano and Candito, 2012), we are not aware of previous work on lexical generalization for out-of-domain dependency parsing.

A first small contribution of our work is the reproduction for French of previous approaches used successfully for English dependency parsing, with to our knowledge the first experiments using HAC clustering and WordNet-based approaches for lexi-

cal generalization applied to French parsing. Otherwise, our primary contributions in this chapter are two-fold: (i) we introduce a multiple assignation strategy for lexical feature replacement in our distributional lexical class spaces, with single assignation having been the standard replacement strategy in previous works on parsing; and (ii) we test the novel use of lexical generalization derived from a bridge corpus to improve out-of-domain dependency parsing.

4.1.2 Framework for Distributional Methods

We base our terminology and methods on the work of Lin (1998), which used word context relations to calculate distributional lexical similarity, and the subsequent work of Curran (2004), which distinguished between weight and measure functions and evaluated different functions on a semantic similarity task for English. The basic framework we use for distributional methods is as follows:

1. Extraction: obtain from a preprocessed text corpus counts of lexical terms and contexts in which they appear, with various possible types and complexity of context.

2. Weight: modify the raw counts in order to better reflect the relevance of a context to a particular lexical term.

3. Measure: calculate the similarity between pairs of lexical terms by representing each as a vector of its weighted context counts.

4. Thesaurus: output a thesaurus with each lexical term having an entry listing the other most similar lexical terms.

More precisely, we first define a *context relation* to be the tuple (w, r, w'), where w is a primary lexical term (we use lemmas) that occurs in a particular context; in our work, contexts consist of a relation r and a secondary lexical term w'. Commonly-used contexts for w include syntactic dependencies, fixed-size windows (such as bigrams), and bag-of-words representations of documents. The choice of context dictates the semantic relationship obtained between primary lexical terms; Agirre et al. (2009) find that syntactic dependencies and fixed-size windows best capture *semantic similarity* (synonymy, antonymy, hyponymy, etc.(while bag-of-words approaches capture broader *semantic relatedness* such as shared topic. We choose to use syntactic dependencies, with r representing a dependency relation and taking values corresponding to the direction of the dependency and its label. In the cases of PP-attachment and coordination, prepositions and coordinating conjunctions are

folded into r; for PP-attachment this means that the pair of lexical terms are the governor of the PP and the PP object, while for coordination this means that the pair of lexical terms are the two conjuncts. As an example, ('aboyer', SUJ, 'chien') is a possible context relation, with 'aboyer' ("to bark") as the primary lexical term that governs the secondary lexical term 'chien' ("dog"). To represent a context relation for the inverted direction of that dependency, we would use ('aboyer', SUJ$'$, 'chien').

After context relations are extracted from an automatically parsed text corpus, each primary lexical term w can be represented as a frequency vector $v^w \in \mathbb{R}^d$, where d is the number of unique contexts appearing in the corpus, and $v^w_i = freq(w, r, w')$, where i corresponds to the context $c_i = (r, w')$. From this point forward, we exclusively use the notation c for a context as we no longer need to refer separately to the relation r and secondary lexical term w'.

Weight and Measure Functions

Term similarity metrics are used to calculate similarities between pairs of primary lexical terms w_1 and w_2 using frequency vectors v^{w_1} and v^{w_2}. Curran (2004) breaks term similarity metrics down into two components: a *weight* function transforms the raw frequency of each context relation by determining the informativeness of the context, while a *measure* function subsequently calculates the similarity between two weighted frequency vectors.

The formulas for the weight and measure functions used in our experiments are listed in Tables 4.2 and 4.3. The symbol $*$ as an argument to a function is shorthand for taking the sum of the function over all possible values for that argument; a pair of subscripted asterisks in the scope of a sum indicates that the variables are bound together, so $\sum wgt(w_1, *_c) \times wgt(w_2, *_c)$ is a sum over all c that are in the set of contexts for both w_1 and w_2. Also, p denotes the probability of a context relation, where $p(w, c)$ is estimated as $f(w, c)/f(*, *, *)$. Finally, wgt denotes the application of some fixed weight function to a context relation count.

The weight functions we consider for a context relation (w, c) are as follows. First is relative frequency (RELFREQ), which simply normalizes the frequency of (w, c) with respect to the frequency of its primary lexical term w. Next is the t-test (TTEST), which computes the difference between observed and expected means of w and c, scaled by the variance of the data; the expected means are computed under the null hypothesis of independence, with a higher t-value leading us toward a rejection of the null hypothesis. Finally is pointwise mutual information (PMI), which is a widely used information theoretic metric comparing two variables, in this case w and c.

RELFREQ	$\frac{f(w,c)}{f(w,*)}$
TTEST	$\frac{p(w,cc)-p(*,c)p(w,*)}{\sqrt{p(w,c)/f(*,*)}}$
PMI	$log\left(\frac{p(w,c)}{p(*,c)p(w,*)}\right)$

Table 4.2: Weight functions for finding context informativeness.

COSINE	$\frac{\sum wgt(w_1,*_c)\times wgt(w_2,*_c)}{\sqrt{\sum wgt(w_1,*)^2\times\sum wgt(w_2,*)^2}}$
JACCARD	$\frac{\sum min(wgt(w_1,*_c),wgt(w_2,*_c))}{\sum max(wgt(w_1,*_c),wgt(w_2,*_c))}$
LIN	$\frac{\sum wgt(w_1,*_c)+wgt(w_2,*_c)}{\sum wgt(w_1,*)+\sum wgt(w_2,*)}$

Table 4.3: Measure functions for calculating distributional similarity.

The measure functions we consider for a pair of primary lexical terms w_1 and w_2 and their respective weighted vectors v^{w_1} and v^{w_2}, are as follows. First is cosine similarity (COSINE), which calculates the cosine of the angle between v^{w_1} and v^{w_2}. Next is the Jaccard measure (JACCARD), which compares the number of common contexts to the number of unique contexts between w_1 and w_2. Finally is the measure that Lin (1998) uses (LIN), which is an information theoretic measure to determine the similarity between w_1 and w_2.

We note that in the work of Curran (2004) many other weight and measure functions were considered. Although we could have tested a wide range of weight and measure functions in our experiments, the intrinsic evaluation of distributional thesauri is not the primary subject of our research. We thus decided instead to use the conclusions of Curran (2004) and restrict our evaluation to a combination of basic measures and measures that they found to be high-performing.

4.1.3 Lexical Class Spaces

We now turn to a discussion of the three lexical class spaces that we have chosen to explore, with the goal of finding appropriate high level classes in which to group word forms so as to reduce lexical data sparseness in features for parsing. These class spaces are those of lemmas, clusters, and semantic senses.

A primary concern regarding the definition of generalized lexical classes for the spaces of lemmas and semantic senses is that a word form can be associated with a list of lexical classes in that space, each with an associated relevance or similarity score. There are thus two ways to map word forms to a particular generalized space: the *single assignation* strategy, which is the standard approach used in the

literature, maps the word form to the single highest scoring lexical class; and the *multiple assignation* strategy, which we will test in our experiments, maps the word form to the top-k scoring lexical classes. Note that it would be possible to use contextual WSD to dynamically alter the lexical scores for a word form in a particular sentence depending on the surrounding sentential context. Although a contextual WSD approach to scoring lexical classes may be preferable, we leave this for future work; in the experiments presented in this chapter, lexical class scores are calculated once over a large corpus and then stored in a fixed resource.

The use of a multiple assignation strategy for integrating generalized lexical classes into parsing models can be achieved by having multiple indicator features fire for a single categorical feature template during parsing. Consider the feature template $\text{LEM}_{\beta[0]}$ for transition-based parsing in Chapter 2. If we choose to use the space of semantic senses, then this template would become $\text{SENSE}_{\beta[0]}$. Our definition of categorical features normally dictates that a single indicator feature pertaining to this template would fire in any given configuration, but we can allow for multiple such features to fire, for instance having both indicator features $\text{SENSE}_{\beta[0]}$='avocado' and $\text{SENSE}_{\beta[0]}$='lawyer' fire for the French word form 'avocat'.

Though it is not immediately clear whether multiple assignation is preferable to single assignation for parsing, we note that data sparseness is reduced at least as much with multiple assignation as it is with single assignation, and most likely even more. On the other hand, having more features fire may lead to slower learning and prediction times. We will return to these and other questions concerning the pros and cons of single assignation and multiple assignation strategies in the experiments section of this chapter.

Given the fact that we allow for the multiple assignation strategy to be employed in our spaces, it turns out that the construction of a lexical resource for each will rely on the use of a distributional thesaurus in some capacity. We thus term them *distributional lexical class spaces* for word forms.

Lemmas with Distributionally Similar Neighbors

In the space of lemmas, the single assignation strategy can be thought of as the simple, widely used baseline of replacing a word form with its lemma. In fact, we have already used this approach throughout our experiments in the preceding chapters as a first step toward reducing lexical data sparseness, as we first discussed in our benchmarking experiment for French in Section 2.1.2. It is thus the multiple assignation strategy that we are interested in for the lemma space in this chapter.

The approach we use to map a word form to multiple lemmas in the lemma space is as follows. We first identify the actual predicted lemma associated with

a word form using a simple deterministic non-contextual lemmatization algorithm, which uses the Lefff resource as its morphological lexicon. Given this predicted lemma, we query a distributional thesaurus for the k-most similar neighboring lemmas, including the original lemma, with the cutoff k left as a tunable parameter. The goal is to balance between including more neighbors so as to reduce data sparseness, and limiting the amount of noisy or irrelevant lemmas included in the mapping. This process gives us a set of lemmas with which to replace the original word form.

In more formal notation, suppose we have a word form w with corresponding predicted lemma l, and a distributional thesaurus over lemmas that can be represented as a similarity function $D(l_1, l_2)$, where $l_1, l_2 \in V$ and V is the vocabulary of lemmas. We define the set of k-most similar lemmas to be $N_k(l)$, with the restrictions that $N_k(l) \subseteq V$ and $l \in N_k(l)$.

Clusters with Hierarchical Agglomerative Clustering

We now describe the space of clusters, whose generalized lexical classes are defined by clustered groupings of distributionally similar lemmas. The literature for clustering approaches in computer science is extensive, and these approaches can be roughly divided into those that produce *hard clustering*, where each original item is a member of a single cluster, or *soft clustering*, where an original item can be a member of multiple clusters. In our experiments we ended up exclusively testing hard clustering, which has been well-studied for different NLP applications and particularly for parsing (Koo et al., 2008; Candito and Crabbé, 2009; Sagae and Gordon, 2009), meaning that for the cluster space we use only the single assignation strategy. It would have been interesting to study soft clustering as well, especially since it would have fit naturally with the multiple assignation strategy; we leave this for future work.

The particular clustering algorithm we choose to use is that of *Hierarchical Agglomerative Clustering (HAC)*. Our following description of this algorithm is based on that appearing in a textbook by Hastie et al. (2008). A first way to view this algorithm is as one in a family of *hierarchical clustering* methods, which produce complete hierarchical representations: at the lowest level each cluster contains a single item, while at the highest level there is only one cluster containing all the data. A hierarchical clustering of a data set can be obtained using methods that are *agglomerative*, meaning bottom-up, or *divisive*, meaning top-down. In the agglomerative strategy we employ, the algorithm starts at the bottom and at each level recursively merges a selected pair of clusters into a single cluster, which produces the grouping at the next higher level with one less cluster. Note that there are then as many levels as there are items from the data set.

The most important decision to make for HAC clustering involves the criteria for merging clusters at each level. A major requirement for this approach is that an item similarity measure $s(i, j)$ exists between pairs of items i, j in the original data set. The natural agglomerative strategy is then to merge the two clusters at the current level with the largest similarity. The cluster similarity measure $S(A, B)$ between pairs of clusters A, B is defined in one of three standard ways:

1. Single linkage, which takes the cluster similarity to be that of the most similar pair of constituent items:

$$S_{SL}(A, B) = \max_{i \in A, j \in B} s(i, j) \tag{4.1}$$

2. Complete linkage, which takes the cluster similarity to be that of the least similar pair of constituent items:

$$S_{CL}(A, B) = \min_{i \in A, j \in B} s(i, j) \tag{4.2}$$

3. Average linkage, which takes the cluster similarity to be the average over that of every pair of constituent items:

$$S_{AL}(A, B) = \frac{1}{|A| \cdot |B|} \sum_{i \in A} \sum_{j \in B} s(i, j) \tag{4.3}$$

When comparing these three cluster similarity measures, one can view single linkage and complete linkage as lying at opposite extremes; single linkage tends to result in clusters with items that are linked by a series of close intermediate items but with potentially low intra-cluster similarity, while complete linkage tends to result in compact clusters with higher intra-cluster similarity but items that may be closer to members of other clusters than they are to some members of their own cluster. Average linkage represents a compromise between the two extremes, attempting to produce relatively compact clusters that are relatively far apart from each other. We choose to use average linkage clustering in our work.

A final aspect of HAC clustering is that, while a full hierarchy is generated during the clustering process, the ideal level of clustering needs to be identified. if there are n original items, then the number of output clusters can be anywhere between 1 and n. We introduce a parameter z to the HAC clustering process, where z is in $[0, 1]$ and is defined as the proportion of output clusters with respect to the number of original items, meaning that there will be zn unique clusters in the output.

From this general description of HAC clustering, it is clear that for our particular application we use lemmas as the items to cluster and the distributional thesaurus provides the necessary item similarity metric. As for the parameter z, this is tested at different values during our experimental parameter tuning in Section 4.2.

WordNet Senses with Automatic Sense Ranking

We now describe the space of semantic senses, with multiple possible senses for a given word form ranked using the technique of automatic sense ranking. We will first provide a description of WordNet and the sense hierarchy used by that resource, and then describe the sense ranking procedure, which uses a distributional thesaurus, and allows us to define a space of semantic senses with a multiple assignation strategy for word form replacement.

The *Princeton WordNet (PWN)* is a manually built resource containing a sense hierarchy for English built through an extensive effort in the 1990's (Fellbaum, 1998). It constitutes a major resource in the field of NLP, having been used in a wide range of applications including word-sense disambiguation, information extraction, machine translation, document classification and text summarization, among others. In summary, the PWN is a broad coverage lexical network of English words. Nouns, verbs, adjectives, and adverbs are each organized into networks of *synonym sets (synsets)*, which represent underlying lexical concepts and are interlinked through a variety of relations. An useful characteristic is that a word with multiple meanings can appear in a different synset for each of its senses. Although our interest in the resource is mostly limited to synsets, a major aspect of the PWN is its hyponymy hierarchy for senses, with relations indicating that a subsumed sense is an instance of a higher sense (e.g. a 'car' is a type of 'vehicle'). Although we do not use this hierarchy directly, it will come into play shortly when we require a measure for determining the similarity between two senses.

Given the relevance and usefulness of the PWN for a variety of applications in NLP, there has been great interest in building comparable resources for other languages. The EuroWordNet project (Vossen, 1998) in the late 1990s ported the existing sense hierarchy from English to other languages, under the assumption that the fundamental sense hierarchy would not be too different for those languages. This made the task of constructing the WordNet resource easier, as the remaining work would just be to assign words in a different language to the appropriate sense. In our work, we use the *French EuroWordNet (FREWN)* portion of the EuroWordNet. In an analysis of the coverage of the FREWN with respect to the PWN (Sagot and Fišer, 2008), after normalizing the synsets to PWN version 2.0, the FREWN was noted as containing 22,121 non-empty synsets and for nouns and verbs, compared to

93,197 synsets for nouns and verbs in the original PWN. It is important to keep in mind, therefore, that the FREWN has a rather sparsely populated sense hierarchy, which might make it a less effective resource for French than the PWN has proven to be for English. Another WordNet-style resource that exists for French is the WOLF (Sagot and Fišer, 2008), which was built semi-automatically; in preliminary experiments we found it to be less useful for parsing than the manually validated FREWN, so we do not include it in our experimental section.

For a particular language with a corresponding WordNet resource, we can take a word form x, or more precisely its lemma form, and query the WordNet resource to obtain a set S_x of senses for the word form. The purpose of *Automatic Sense Ranking (ASR)* is to rank the prevalence of each such sense with respect to the word form, using a distributional thesaurus to carry out this task. We are interested in ASR because it provides a natural way of identifying a highest-ranked sense for the purposes of single assignation replacement, which is an approach that has been previously used by Agirre et al. (2011) for English parsing. In addition, for the multiple assignation strategy it may be useful to restrict the number of replacement senses to the k-most prevalent ones, in order to discard infrequent senses that would introduce noise into the feature vectors for parser model learning.

The approach we use for ASR follows that of McCarthy et al. (2004). Representing the distributional thesaurus as a similarity function $D(x, y)$ over lemmas, letting $N_x(k)$ be the set of k-nearest neighbors for x in the thesaurus, and letting $W(s_1, s_2)$ be a similarity function over a pair of synsets in WordNet, we define a prevalence metric $R_x(s)$ as follows:

$$R_x(s) = \sum_{y \in N_x(k)} D(x, y) \frac{\max_{s' \in S_y} W(s, s')}{\sum_{t \in S_x} \max_{s' \in S_y} W(t, s')} \tag{4.4}$$

This metric essentially weights the semantic contribution of each distributionally similar neighbor $y \in N_x(k)$ toward a given sense s of x. For each neighbor $y \in N_x(k)$, its contribution is calculated as follows: (i) the similarity between s and the sense s' of y that is most similar to s is computed; (ii) that similarity is normalized by corresponding similarities between all senses t of x and the senses of y that are most similar to each such t; and (iii) the resulting normalized score is further weighted according to the distributional similarity $D(x, y)$ between x and the current neighbor y.

The definition of a similarity function over a pair of synsets has been studied in the WordNet literature, with a number of approaches being identified; the function can use the path along the sense hierarchy linking the two synsets, or even measure

the lexical overlap in the glosses of the synsets. The particular synset similarity function we use is that of Jiang and Conrath (1997), which is the one used in the original work of McCarthy et al. (2004). It defines a distance measure between two synsets as a function of their information content, as well as the information content of their most informative (or most specific) superordinate synset in the sense hierarchy. In order to calculate the information content of a synset, outside textual corpus data is required to populate synsets in the WordNet hierarchy with frequency counts; these counts allow for the probability of a synset $p(s)$ to be estimated, from which the information content is calculated as $\text{IC}(s) = -log(p(s))$.

4.2 Lexical Class Experiments

In this section we describe two sets of experiments related to generalized lexical classes. The first set of experiments evaluates our distributional methods intrinsically, independently of their application to parsing, in order to ascertain which combination of weight and measure functions for similarity are preferable. This is achieved by comparing different generated distributional thesauri against FREWN synsets, which we use as a gold standard. The second set of experiments then uses the preferred distributional thesaurus to generate resources for our three distributional lexical class spaces, and evaluates the efficacy of modifying lexical features with these resources in dependency parsing and correction systems (as presented in Chapters 2 and 3) for both in-domain and out-of-domain parsing.

4.2.1 Methods and Setup

We now present the setup of our distributional thesaurus and generalized lexical class experiments. For all methodological details concerning parsing with ARC-EAGER and neighborhood correction, see our previous descriptions in Section 2.3 and Section 3.3, respectively. As was the case for our previous experiments, we used our own Python implementation of the algorithms and methods from this chapter; we reiterate that we plan to release a package of the code used in this thesis following its publication. Our methodology for experiments in lexical generalization involves the following elements: constructing distributional thesauri from an automatically-parsed external corpus; preprocessing the FREWN for use in our experiments; intrinsically evaluating of our thesauri against the FREWN gold standard; constructing resources for our three distributional lexical class spaces from a distributional thesaurus; modifying feature vectors using the single assignation and multiple assignation strategies during parsing and correction; and finally modifying

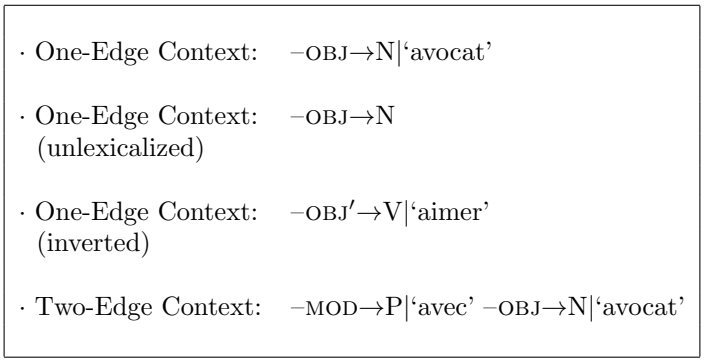


Figure 4.1: Example dependency contexts for the verb lemma 'manger'. The first couple of one-edge contexts correspond to the sentence "Jean mange un avocat" ("Jean eats an avocado"), the inverted one-edge context corresponds to the sentence "Jean aime manger" ("Jean loves to eat"), and the two-edge context corresponds to the sentence "Jean mange avec un avocat" ("Jean eats with a lawyer").

the in-domain lexical generalization experiments for domain adaption in the medical domain.

Construction of Distributional Thesauri

Our distributional thesauri were, by design, restricted to the two main largest open-class POS categories: nouns and verbs. Within a thesaurus, an additional restriction we imposed was that each lexical term would only have neighbors of the same POS category. It should be noted that the remaining open class POS categories, adjectives and adverbs, are potentially interesting POS categories to include; however, from early testing we observed that generalized lexical classes over adjectives and adverbs were not very useful for parsing.

The external French text corpora we considered as the basis for generating our distributional thesauri were the same as those used in Chapter 3 for self-trained two-stage parsing experiments. Of the two sets in the journalistic domain used in those experiments, we used here only the AFP corpus. We found in the earlier self-training experiments that the AFP corpus appeared to be closer and more relevant to the FTBDep than the ER corpus, with better self-training results when using AFP compared to ER. As before, the corpus was preprocessed using the Bonsai tool, which performed sentence segmentation, word tokenization, and additionally, consistent with our automatic annotation of the FTBDep in Section 2.3, performed automatic POS tagging with the MElt package and lemmatization with the Lefff lexicon.

We now turn to details concerning the generation of distributional thesauri

from the preprocessed external corpora. First, *syntactic contexts* for each lemma were extracted from the corpus. As a general policy, we used all syntactic dependencies in which the secondary lemma had an open-class POS tag, with labels included in the contexts and two-edge dependencies used in the case of prepositional-phrase attachment and coordination. For verb lemmas, we decided to further limited contexts to those dependencies in which the verb was governor, and we also added unlexicalized versions of contexts to account for subcategorization. Example contexts are shown in Figure 4.1. After gathering the context relations (primary term and context), we filtered out those for which either the primary term or the context did not appear in at least 500 context relations. This resulted in a vocabulary size for our thesauri of 6,682 noun lemmas and 2,355 verb lemmas.

Each pair of primary term lemma w and context c was subsequently weighted for relevance or informativeness using one of three weight functions, and the distributional similarity between pairs of primary term lemmas w_1, w_2 was calculated using one of three similarity measure functions. This resulted in nine different thesaurus settings to be tested in our preliminary intrinsic evaluation, with the best one to be used later in the creation of lexical class resources.

Preprocessing the FREWN

An important step in our experimental setup was to convert the FREWN from its original version 1.5 to the PWN version 3.0, as most available tools for working with WordNet resources require the version 3.0 sense hierarchy.

In order to achieve this, we found a WordNet synset mapping tool[1] capable of performing this conversion. After discarding a small number of synsets from the FREWN that were not covered by the mapping, we were able to retain entries for 9,833 noun lemmas and 2,220 verb lemmas. Subsequent references to the FREWN will refer to this converted version with the PWN version 3.0 sense hierarchy.

Intrinsic Evaluation of Thesauri

In our intrinsic evaluation of distributional thesauri, we followed the idea of Lin (1998) in comparing each of our automatically constructed thesauri against a gold-standard WordNet resource. Since we are working with French, we used the FREWN as our gold standard reference. This initial evaluation was actually performed in an earlier experiment that we conducted for French (Henestroza Anguiano and Denis, 2011), though in that experiment the ER corpus was used instead of the AFP corpus to create the distributional thesauri.

[1] http://nlp.lsi.upc.edu/tools/download-map.php

Top Noun Measures		**Top Verb Measures**	
Setting	INVR	**Setting**	INVR
PMI, COSINE	0.281	PMI, COSINE	0.334
TTEST, COSINE	0.266	TTEST, COSINE	0.332
TTEST, JACCARD	0.260	TTEST, JACCARD	0.330
PMI, JACCARD	0.259	PMI, JACCARD	0.312
...		...	

Table 4.4: Average INVR evaluation scores for the top distributional thesauri by POS category. Each setting consists of a particular weight function (out of RELFREQ, TTEST, or PMI) and measure function (out of COSINE, JACCARD, or LIN).

For our evaluation metric, that experiment used *average inverse rank (INVR)*, a standard information retrieval metric. For each term w, we considered all terms appearing in a synset with w in the WordNet reference to be relevant, while other terms were considered irrelevant. In the distributional thesaurus to be evaluated, the entry for w was thus taken as a ranked list of query results (neighbor terms ranked by descending similarity). The INVR metric returned the sum, over relevant neighboring terms, of the inverse of that term's rank in the list. The average INVR was taken over all terms to be evaluated, providing an evaluation metric for the quality of a distributional thesaurus.

All nine possible combinations of weight and measure functions were evaluated for distributional thesauri over both nouns and verbs, and the results are reported in Table 4.4. A first clear observation is that the RELFREQ weight function and the LIN measure function do not perform very well, with results lower than those for the top performing settings appearing in the table. Of the top performing settings, we found that a combination of the PMI weight function and the COSINE measure function produced the highest quality thesaurus with respect to the FREWN reference. Consequently, we exclusively used this thesaurus setting for the remainder of the experiments presented in this chapter.

It is important to note that high performance for a particular setting in this intrinsic evaluation does not necessarily translate to high performance when using a thesaurus for lexical generalization during parsing. On the other hand, considering all nine possible thesauri for our parsing experiments was not possible given reasonable computational time constraints, as well as all the other parameters and settings that needed to be tuned and evaluated. Therefore, we used this intrinsic evaluation as a method for obtaining the thesaurus setting that would give us the best chance of applying lexical generalization effectively in our parsing models.

Construction of Lexical Class Resources

We now discuss the creation of lexical class resources from a distributional thesaurus for our three spaces: lemmas, clusters, and semantic senses.

For the space of lemmas, the resource was simply the distributional thesaurus itself. For the single assignation strategy, we needed only to use the predicted lemma for a given word form, and for the multiple assignation strategy we needed only to identify its k-most similar neighboring lemmas according to the thesaurus.

For the space of clusters, as noted earlier we chose to use the HAC clustering algorithm, which is suitable for creating hard clusters given pair-wise distance or similarity measures between basic items. We used the average-linkage metric for cluster agglomeration, and tested varying levels of clustering, with the parameter z determining the proportion to set for cluster vocabulary size compared to the original lemma vocabulary size. These clusters then constituted the resource for the cluster space, which supports only the single assignation strategy and replaces each word form with its assigned cluster identifier.

Finally, for the space of semantic senses, as noted earlier we chose to use WordNet ASR to rank senses from the FREWN for each word form. We used NLTK, the Natural Language Toolkit (Bird et al., 2009), to calculate similarity between synsets. As explained earlier, our method for performing ASR follows that of McCarthy et al. (2004). We used k=16 for the distributional k-most similar neighboring lemmas to consider when ranking the senses for a lemma,[1] and we used the synset similarity function of Jiang and Conrath (1997), with default information content counts from NLTK calculated over the British National Corpus[2]. It is important to note that there was understandably not a complete overlap between the vocabularies for the FREWN and our distributional thesaurus. As noted earlier, our thesaurus contains 6,682 noun lemmas and 2,355 verb lemmas, while the FREWN has entries for 9,833 noun lemmas and 2,220 verb lemmas. Of the FREWN lemmas, only 4,212 nouns and 1,478 verbs also appeared in the thesaurus; for each FREWN lemma not covered by the thesaurus, we simply allocated equal rank to all of its senses.

Lexical Modification of Feature Vectors

The process for modifying feature vectors during dependency parsing and correction is relatively straightforward, as noted in Section 4.1.3. When any of the three

[1] McCarthy et al. (2004) used k=50, though they noted that with a value as low as k=10 the results did not change significantly. We used k=16 for convenience, since otherwise in our experiments we never looked at more than 16 neighbors for a lemma.

[2] `http://www.natcorp.ox.ac.uk/`

distributional lexical class spaces is invoked, lexical feature templates are replaced with new templates corresponding to the new space, if needed. For instance, the template $\text{LEM}_{\beta[0]}$ would become $\text{SENSE}_{\beta[0]}$ if we are invoking the semantic sense space.

Additionally, when using the multiple assignation strategy during learning and prediction, each feature template is allowed to have more than one of its constituent indicator features fire simultaneously. For instance, if the word form (and lemma) 'avocat' has two ranked senses corresponding to 'avocado' and 'lawyer', then the following two indicator features would fire when 'avocat' is encountered: $\text{SENSE}_{\beta[0]}$='avocado' and $\text{SENSE}_{\beta[0]}$='lawyer'.

Note that we do not incorporate the lexical class scores themselves into the feature vectors. We actually did explore this option in a previously published work (Henestroza Anguiano and Candito, 2012), in which we converted the lexical class scores into a probabilistic value and assigned these as the values for lexical features, following the generalized approach for learning with probabilistic features proposed by Bunescu (2008). We ultimately chose not to replicate that decision in our present experiments for two reasons: (i) it breaks the 0-1 indicator feature paradigm, which we have used until now throughout our experiment, potentiallt affecting the model optimization process in unknown ways; and (ii) it is unclear whether distributional similarity scores in the lemma space or prevalence scores in the semantic sense space should be viewed probabilistically, as they arise from metrics that have been chosen for their ability to *rank* items. It would nonetheless be interesting to compare the probabilistic approach to the one we took here; we leave this for future work.

Domain Adaptation Setup

In addition to our primary experiments on using lexical generalization to improve parsing within a single domain, given that the FTBDep and AFP both contain text in the journalistic domain, we required additional setup to test domain adaptation for parsing in the biomedical domain.

As in Chapter 3, we used the EMEA development and test sets from the Sequoia annotated corpus (Candito and Seddah, 2012a) as our reference sets. For the experiments in this chapter, we additionally used the approach of employing a *bridge corpus*, which contains text from both a source and a target domain, as in a previous study for domain adaptation in phrase-structure French parsing (Candito et al., 2011). The hypothesis behind this approach is that if we build lexical resources from a large corpus containing both journalistic and biomedical text, the resulting lexical classes will be able to bridge the lexical gap between the two domains; given that our training data for parsing lies exclusively within the source journalistic domain, using generalized lexical classes as features may result in better parsing

of sentences in the target domain, which may contain word forms that appear less frequently in the source domain but share a lexical class with one or more word forms that do appear regularly in the source domain.

In order to build our bridge corpus, we again, as in Chapter 3, used the EMEA unlabeled set obtained from previous work on preprocessing the EMEA corpus for French (Candito et al., 2011). Our bridge then consisted of the approximately 4750k sentences from the AFP corpus, representing the journalistic domain, and the approximately 265k sentences from the EMEA unlabeled set, representing the biomedical domain. Due to the fact that the EMEA unlabeled set is much smaller than the AFP corpus, coupled with the fact that our distributional methods require a minimum frequency cutoff for lemmas appearing in the input corpus in order to make the thesaurus construction process tractable, we up-weighted the EMEA unlabeled set to be eighteen times its original size in order to match the size of the AFP corpus. Given this bridge corpus, the methodology for constructing lexical resources and parsing with them was the same as previously described for the in-domain experiments.

4.2.2 Results

This section describes the results of our lexical generalization experiments, where we used transition-based parsing and neighborhood correction as the base machine learning approaches over which feature replacement with generalized lexical classes is evaluated. We first present results on standard in-domain parsing when using each of the three lexical class spaces discussed in this chapter: distributionally similar lemmas, HAC clusters, and WordNet senses ranked with ASR. We then present corresponding results on domain adaptation for parsing, with a bridge corpus containing both journalistic and biomedical text replacing the basic journalistic corpus as the basis for our distributional methods when constructing lexical resources. For domain adaptation, we also used development and test sets from the biomedical domain to evaluate whether the bridge corpus approach can improve parsing performance on out-of-domain text.

In-Domain Parsing with Lexical Classes

Table 4.5 shows our results on the FTBDep development set when using the lexical generalization feature replacement approach for parsing in-domain. Although the differences in performance compared to the baseline were slight, we can observe certain trends. For the lemma space, adding more k-nearest lemmas to the multiple mapping resulted in worse performance; the best setting, which is slightly better

	Settings	LAS	UAS
Baseline	ARC-EAGER	86.4	89.1
Lemma Space	POS=N, k=2	86.6	89.2
	POS=N, k=4	86.5	89.2
	POS=N, k=8	86.0	88.7
	POS=V, k=2	86.5	89.1
	POS=V, k=4	86.4	89.1
	POS=V, k=8	85.5	88.4
Cluster Space	POS=N, z=0.2	86.5	89.1
	POS=N, z=0.4	86.5	89.2
	POS=N, z=0.6	86.6	89.2
	POS=N, z=0.8	86.7	89.3
	POS=V, z=0.2	86.4	89.2
	POS=V, z=0.4	86.5	89.2
	POS=V, z=0.6	86.6	89.2
	POS=V, z=0.8	86.5	89.2
Sense Space	POS=N, k=1	86.4	89.1
	POS=N, k=2	86.4	89.2
	POS=N, k=4	86.4	89.1
	POS=N, k=8	86.4	89.1
	POS=V, k=1	86.4	89.1
	POS=V, k=2	86.6	89.2
	POS=V, k=4	86.3	89.0
	POS=V, k=8	86.3	89.0

Table 4.5: LAS and UAS results, in percent, over the FTBDep development set when using in-domain lexical generalization parsing approaches. Results are grouped into the baseline system, lemma space systems with varying POS and k-nearest lemmas used, cluster space systems with varying POS and z cluster vocabulary proportions used, and sense space systems with varying POS and k-highest ranked senses used.

than the baseline, occured when we used k=2, meaning that we replace each lexical feature with two corresponding to the original lemma and its nearest neighbor from the distributional thesaurus. For the cluster space, the best value for z varied depending on the POS category used for lexical generalization; z=0.8 worked best for nouns, while z=0.6 worked best for verbs. Finally, for the sense space, replacing each lemma with up to k=2 highest ASR senses worked best for both nouns and for verbs; as was the case in the lemma space, using higher k resulted in worse performance. Overall, we can observe that the cluster space lead to the highest parsing improvement, though the differences in LAS and UAS were not statistically significant.

Given these results on the development set, we took the best settings and evaluated them on the FTBDep test set. For this final parsing evaluation, we combined noun and verb lexical generalization, giving us one combined setting for each

	Settings	LAS	UAS
Baseline	ARC-EAGER	87.1	89.7
	+CORRECTION	87.5	90.2
Lemma Space	POS=N&V, k_N=2, k_V=2	87.4*	89.9
	+CORRECTION	87.8*	90.3
	+CORRECTION_LEXGEN	87.8*	90.3
Cluster Space	POS=N&V, z_N=0.8, z_V=0.6	87.2	89.8
Sense Space	POS=N&V, k_N=2, k_V=2	87.2	89.9

Table 4.6: LAS and UAS results, in percent, over the FTBDep test set when using in-domain lexical generalization parsing and correction approaches. Results are grouped into the baseline systems, lemma space systems cluster space systems, and sense space systems. Each lexical generalization system contains either a parser alone, a parser with a new corrector that uses no lexical generalization (+CORRECTION), or a parser with a new corrector that uses the same lexical generalization approach as its corresponding parser (+CORRECTION_LEXGEN). $*$ indicates a statistically significant improvement over the baseline, with approaches without correction compared to the baseline without correction, and those with correction compared to the baseline with correction.

of the three lexical generalization spaces. In addition to testing these settings for parsing, for the resulting best setting (lemma space) we re-trained new neighborhood correctors optimized to correct errors made by that particular parser (cf. the jack-knifing training procedure described in Chapter 3). The neighborhood correctors were trained in two different ways: using no lexical generalization, or using the same lexical generalization setting as the parser on whose errors it was trained. The results for the final evaluations for lexical generalization in ARC-EAGER parsers and neighborhood correctors on the FTBDep test set are shown in Table 4.6.

These final results on the FTBDep test set again show only modest improvements over the baseline when using lexical generalization, but some of them are nonetheless statistically significant. The best parser using lexical generalization in the lemma space achieved a higher LAS (87.4) with a statistically significant difference compared to the baseline ARC-EAGER parser (87.1). For that same setting, a newly trained neighborhood correction model with or without lexical generalization lead to a two-stage parsing system that had a higher LAS (87.8) with a statistically significant difference compared to the baseline two-stage parsing system (87.5). Unfortunately, the use of lexical generalization in the correction model did not provide further improvement in parsing accuracy.

Domain Adaptation for Parsing with Lexical Classes

We now turn to our evaluation of domain adaptation for parsing with lexical classes, with lexical resources constructed using a bridge corpus consisting of journalistic text

	Settings	FTBDep dev		EMEA dev	
		LAS	**UAS**	**LAS**	**UAS**
Baseline	ARC-EAGER	86.4	89.1	84.2	87.0
Lemma Space	POS=N, k=2	86.6	89.2	84.2	87.0
	POS=N, k=4	86.5	89.2	83.9	86.7
	POS=N, k=8	86.0	88.8	83.1	86.0
	POS=V, k=2	86.5	89.2	83.9	86.8
	POS=V, k=4	86.5	89.1	84.0	86.7
	POS=V, k=8	86.0	88.8	83.3	86.1
Cluster Space	POS=N, z=0.2	86.4	89.0	84.2	87.0
	POS=N, z=0.4	86.5	89.2	84.2	87.0
	POS=N, z=0.6	86.6	89.2	84.3	87.1
	POS=N, z=0.8	86.6	89.2	84.2	86.9
	POS=V, z=0.2	86.4	89.1	84.1	86.9
	POS=V, z=0.4	86.3	89.1	84.1	86.8
	POS=V, z=0.6	86.4	89.1	84.1	86.9
	POS=V, z=0.8	86.5	89.1	83.9	86.7

Table 4.7: LAS and UAS results, in percent, over the FTBDep and EMEA development sets when using bridge lexical generalization parsing approaches. Results are grouped into the baseline system, lemma space systems with varying POS and k-nearest lemmas used, and cluster space systems with varying POS and z cluster vocabulary proportions used.

from the AFP corpus and biomedical text from the EMEA unlabeled set. These results are presented in a similar way to those for in-domain parsing, with the exception that the WordNet sense space was not used. Because for that space the lemma vocabulary is restricted to those lemmas appearing in the FREWN and the synsets are fixed, we hypothesized that the two domain vocabularies would not necessarily be bridged when using the WordNet sense space. For this experiment, the EMEA development and test sets were used in addition to those of the FTBDep in order to determine how well the bridge approach could improve out-of-domain parsing.

Table 4.7 shows our results on the FTBDep and EMEA development sets when using the lexical generalization feature replacement approach for parsing out-of-domain. As for the in-domain evaluation, we note only modest improvements over the baseline for some settings. For the lemma space, as before, adding more k-nearest lemmas to the multiple mapping resulted in worse performance; the best setting occured when we used k=2 for nouns, while no value of k for verbs lead to an improvement on the EMEA development set. For the cluster space, as before, the best value for z varied depending on the POS category used for lexical generalization; z=0.6 worked best for nouns, while no value of z for verbs lead to an improvement on the EMEA development set.

Given these results on the development set, we again took the best settings

	Settings	FTBDep test		EMEA test	
		LAS	UAS	LAS	UAS
Baseline	ARC-EAGER	87.1	89.7	85.9	88.1
	+CORRECTION	87.5	90.2	86.0	88.3
Lemma Space	POS=N, k=2	87.2	89.8	86.1	88.2
	+CORRECTION	87.7	90.3	86.4*	88.5
	+CORRECTION_LEXGEN	87.5	90.2	85.8	87.9
Cluster Space	POS=N, z=0.6	87.2	89.9	85.8	87.9

Table 4.8: LAS and UAS results, in percent, over the FTBDep and EMEA test sets when using bridge lexical generalization parsing and correction approaches. Results are grouped into the baseline systems, lemma space systems cluster space systems, and sense space systems. Each lexical generalization system contains either a parser alone, a parser with a new corrector that uses no lexical generalization (+CORRECTION), or a parser with a new corrector that uses the same lexical generalization approach as its corresponding parser (+CORRECTION_LEXGEN). $*$ indicates a statistically significant improvement over the baseline, with approaches without correction compared to the baseline without correction, and those with correction compared to the baseline with correction.

and evaluated them on the FTBDep and EMEA test sets. For this final evaluation we only used noun lexical generalization, since the results on the development set indicated that verb lexical generalization was not helpful for bridging domains. We thus used one optimal setting for each of the two lexical generalization spaces tested. In addition to testing these settings for parsing, for the resulting best setting (lemma space) we again re-trained new neighborhood correctors optimized to correct errors made by that particular parser. The neighborhood correctors were trained in two different ways: using no lexical generalization, or using the same lexical generalization setting as the parser on whose errors it was trained. The results of these final evaluations for bridge lexical generalization in ARC-EAGER parsers and neighborhood correctors on the FTBDep and EMEA test sets are shown in Table 4.8.

The final results on the FTBDep and EMEA test sets again show only modest improvement over the baseline when using lexical generalization, but the improvement is actually statistically significant when comparing two-stage parsing systems on the EMEA test set. Specifically, the lemma space two-stage parser, with the new corrector not trained with lexical generalization, obtained a higher LAS (86.4) with a statistically significant difference compared to the baseline two-stage parser (86.0). Additionally, this setting had a slightly though not significantly higher LAS (87.7) compared to the baseline two-stage parser (87.5) when evaluating on the FTBDep test set. These results modestly support the hypothesis that a bridge approach for lexical generalization can improve both in-domain and out-of-domain dependency parsing, which would be a nice finding: domain adaptation techniques typically degrade results for in-domain parsing, as was the case in our earlier self-training

experiment (cf. Section 3.3.2). Once again, as with the in-domain lexical generalization experiment, training the neighborhood corrector with generalized lexical classes did not help. The bridge lexical generalization ended up introducing more errors than were corrected, which is a surprising result; we aren't sure why lexical generalization was counter-productive in this case.

Evaluation Summary

Taking our evaluation results as a whole, we can report that both our in-domain experiments using a journalistic corpus and our domain adaptation experiments using a bridge corpus produced lexical generalization models capable of parsing and correction both in-domain and out-of-domain data slightly better than the baseline one-stage and two-stage parsing systems. In each of these cases, the best lexical generalization approach was the one that perhaps least altered the lexical space: it remained in the lemma space, and used multiple assignation to expand each lexical feature into two new ones corresponding to the original lemma and an additional highly similar one.

Although these results are heartening, the parsing improvements we obtain are nonetheless very modest, which calls into question the usefulness of introducing an additional layer of processing to the parsing and correcting models for relatively little gain. Additionally, one of the goals of this work was to finally replicate for French the substantial parsing improvements from clustering obtained by Koo et al. (2008) for English, but we were unfortunately not able to show a significant improvement from clustering. Likewise, we were not able to replicate for French the significant parsing improvements obtained for English by Agirre et al. (2008), in their setting that replaced word forms with the most prevalent WordNet sense obtained using ASR. For the WordNet approach, we note that the quality and coverage of the EWNFR are understandably much lower than those of the PWN, which may have reduced the effectiveness of using senses as lexical classes for French. It is interesting to note that, to our knowledge, no work has successfully used generalized lexical classes to significantly improve dependency parsing for a language other than English.

Chapter 5

Parsing with PP-Attachment Preferences

A likely impossibility is always preferable to an unconvincing possibility.

— *Aristotle*

In this final chapter, we close the second main thread of our thesis previously introduced in Chapter 4, which involves the use of automatically-built lexical resources to improve dependency parsing, with the parsing and correction algorithms from previous chapters serving as a backbone. While Chapter 4 investigated different types of distributional lexical classes that can be used in place of word forms to provide a level of generalization for features during parsing and correction, here we borrow from methods in distributional lexical semantics to model lexical preference for prepositional phrase (PP-) attachment to learn semi-supervised correction models. We focus on PP-attachment because it is an attachment type that is known to be difficult to disambiguate syntactically, and that we have previously identified as contributing to a large portion of parsing errors.

In Section 5.1, we start by motivating our investigation of *lexical preference* for the problem of PP-attachment and discussing the ways in which lexical preference has been used in the literature for parsing and other NLP applications. We then present an overview of the types of lexical preference we consider for PP-attachment, distinguishing between different levels of lexical specificity in the dependent PP; we consider a less-specified level in line with the notion of subcategorization, with the object of the PP generalized to a POS class, and a more specified level with the lemma included for the object of the PP. We then turn to a discussion of the two statistical metrics we use for obtaining preference scores from a large automatically-parsed corpus: (i) a traditionally-used pointwise mutual information metric; and (ii) a novel neighborhood-based relative frequency metric that uses information concerning the neighborhoods of candidate governors for dependents within a predicted parse tree.

In Section 5.2 we then present neighborhood correction experiments for French in which we test the addition of preference features for PP-attachment. This includes an evaluation of the two preference types, subcategorization and lexical assocation preference, as well as the two distributional metrics for obtaining preference scores, pointwise mutual information and neighborhood relative frequency. As was the case for our experiments in Chapter 4, the parsing approaches used as the basis for our experiments are those we previously investigated in Chapters 2 and 3.

5.1 Methods for Lexical Preference

Recalling some essential syntactic notions and ideas we first introduced in Chapter 1, there is an important relationship to consider between, on one hand, syntactic and semantic requirements in a language, and on the other hand, ambiguity in acceptable interpretations for sentences in that language. Suppose we were presented

with a sequence of French POS categories as follows: DET N_1 V DET N_2 P DET N_3. In determining which governor should be assigned to P, we can clearly rule out N_1 due to syntactic constraints; however, both V and N_2 remain syntactically correct governors given this information. While more syntactic information could help (e.g. subcategorization information for V), there are many examples in which artificial syntactic ambiguities would persist and the correct governor could only be determined using semantics. One need only look at the syntax of *modifiers*, or non-subcategorized dependents that are constrained very little by their governor and whose attachment is determined by semantic criteria. Consider the French modifiers "à l'orange" and "à midi" in the sentence "Paul a mangé un canard à l'orange à midi" ("Paul ate a duck à l'orange at noon"): semantics is what allows us to determine that the duck is à l'orange and the eating took place at noon.

The resolution of this artificial syntactic ambiguity (cf. Section 1.2.3), is a primary difficulty to overcome in the task of syntactic parsing, where one typically needs to identify a single 'best' syntactic interpretation for a sentence. Approaches for syntactic parsing need to be able to model, directly or indirectly, the semantic requirements and preferences of a language to correctly parse linguistic constructions that would be ambiguous if semantics were not considered. While statistical approaches for parsing are used precisely because of their potential ability to automatically model requirements observed in a training corpus, data sparseness due to the small size of manually-annotated treebanks is a large impediment. And it is worse for semantic requirements than for syntactic ones, as the former are more dependent on lexical information.

While the lexical generalization approaches we presented in Chapter 4 were aimed at reducing data sparseness in parsing, a downside is that generalizing lexemes might actually worsen the modeling of lexically-based semantic requirements. Our goal here is quite different, as we are not seeking to reduce data sparseness in a training corpus but rather to improve the modeling of lexicalized syntactic and semantic requirements and preferences in a parsing model, using lexical preference information calculated statistically over a large automatically-parsed corpus. We focus our investigation on the problem of PP-attachment, and our reasons for considering only PP-attachment are as follows: (i) it is a prototypical attachment type that causes high levels of artificial syntactic ambiguity; (ii) it accounts for a large number of parsing errors; and (iii) a PP can be either argument or modifier, allowing us to investigate lexical preference models that cover at the same time both syntactic and semantic requirements, which for instance would not be true of nominal objects. It is interesting to note that when discussing the validation of syntactic parses in the FTB, Abeillé et al. (2003) identify PP-attachment as the source of

many difficult annotation cases in which a thorough semantic understanding of the sentence was needed to resolve ambiguity.

In Section 5.1.1 we provide an overview of related work on the use of lexical preference to improve parsing. We subsequently present in Section 5.1.2 the two types of lexical preference we consider for improving parsing, each with a different amount of lexical specificity in the dependent PP position, and we explain why they are relevant to disambiguating PP-attachment. Finally, in Section 5.1.3 we discuss the statistical metrics we use to calculate preference scores: pointwise mutual information (PMI) and neighborhood relative frequency (NRF).

5.1.1 Related Work

Lexical preference is an area that has been studied extensively in NLP, and it is often tied closely to lexical generalization. The techniques for obtaining measures of lexical preference are often related to distributional techniques for building lexical classes through lexical generalization; research tends to focus on statistical metrics of association derived from frequencies of lexical terms in relation to one another in large text corpora. A common area of application for studies on lexical preference is PP-attachment, which as we noted in Section 3.1.1 is typically evaluated as an isolated classification problem outside of an actual parser; we will note which related works evaluated their approaches within a parser.

Semantic Classes and Selectional Preference

It should be noted that a similar area of research is that of identifying what is traditionally termed *selectional preference*, or relations between words according to semantic class, with an application focus on improving word sense disambiguation. This area has been studied extensively, with an important early work being that of Resnik (1992). who first proposed a generalization of lexical association techniques for the statistical discovery of facts involving word classes rather than individual words. Most subsequent work in this line of research has focused on determining selectional preference using some combination of semantic resources for English like SemCor (Miller et al., 1993) and WordNet. To mention just a few works: A study by Agirre and Martinez (2001) places verbs and their arguments into semantic classes based on shared positions; McCarthy and Carroll (2003) investigate the unsupervised acquisition of selectional preference; Gamallo et al. (2005) use an unsupervised strategy to cluster syntactic positions based on shared semantic requirements and evaluates it on a stand-alone PP-attachment problem; Erk (2007) proposes a simple model for the automatic induction of selectional preferences using corpus-based se-

mantic similarity metrics and focusing on the task of semantic role labeling; Bergsma et al. (2008) use a discriminative method for learning selectional preferences from unlabeled text, with positive examples taken from observed predicate-argument pairs and negatives constructed from unobserved combinations; and even more recently, both Ó Séaghdha (2010) and Ritter et al. (2010) explore the use of Latent Dirichlet Allocation to induce topic models for selectional preference.

Our own research goals are slightly different, however, as we try to model direct relations between governors and dependents using only POS categories and lemmas; we try to use a large amount of textual data so as to implicitly capture semantic relations, bypassing the need for the identification of semantic classes in different syntactic positions.

Semi-Supervised Lexical Approaches

The line of research most relevant to ours uses semi-supervised approaches over large syntactically-parsed corpora to model direct lexical relationships. In one of the earliest works of this kind, Briscoe and Carroll (1997) developed a system capable of distinguishing a large number of verbal subcategorization classes, returning relative frequencies for each frame found for each verb. Their approach used a shallow statistical parser over a large corpus, followed by a subcategorization class classifier and the estimation of the probability of membership of these classes.

In a follow up work, Carroll et al. (1998) incorporated subcategorization information obtained using that method into a statistical LR parser for English. In their approach, first the parser proposed a number of syntactic derivations of a sentence, each with a corresponding probability. Then within each derivation, each verb instance and observed subcategorization was identified, and a new 'score' for the derivation was obtained by taking the product of the probability of the derivation and the verb subcategorization probabilities according to the built subcategorization resource. This effectively made subcategorization information part of a post-processing step for disambiguating complete parses, and the authors obtained a significant improvement in precision for verb-argument relations.

Subsequent works have followed in a similar path, with approaches focused more on using lexical preference scores as features within a classification setting. Van Noord (2007) reported parsing improvements for Dutch when using features based on PMI, extracted from a large automatically-parsed external text corpus, and incorporated into a maximum entropy classifier that disambiguated Dutch parses in the HPSG phrase-structure format. They used one feature $z(tag, label)$ for each $(tag, label)$ pair, with tag being the POS category of a governor, and $label$ being a dependency label. For a given candidate dependency parse, the value of a

$z(tag, label)$ feature was the sum of the PMI values for pairs of words that appeared in a dependency matching the $(tag, label)$ constraint.

In another recent study, Zhou et al. (2011) explored the benefits of using web-derived bi-lexical statistics for English dependency parsing. Using frequencies provided by Google hits and Google n-grams, they derived PMI scores for pairs of lexical items and triples of lexical items; for a preposition, the additional lexical items were its object and its governor. They then used these scores as features in a dependency parser implementing a second-order graph based parser (McDonald et al., 2005), and obtained improvements in LAS and UAS performance on the Penn Treebank test sections as well as on out-of-domain text.

Finally, the work that is most similar to ours is that of Mirroshandel et al. (2012), who worked on French dependency parsing with an efficient implementation (Bohnet, 2010) of a second order graph based parser (McDonald et al., 2005). In their approach, bi-lexical affinities were calculated using a metric similar to PMI over an automatically-parsed corpus according to various lexico-syntactic configurations representing verb-argument pairs, coordination, and PP-attachment; for PP-attachment, only the 'de' and 'à' prepositions were covered. A significant improvement in LAS and UAS on the FTBDep was achieved through a post-processing method over an n-best list of parses, where each dependent matching a configuration was reassigned a governor with the highest affinity from among those proposed in the n-best list, followed by a second parsing pass to ensure consistency in the final tree. Our work was conducted in parallel and there is some overlap, but a number of differences exist between the two approaches: we use a linear-time transition-based parser with neighborhood correction as opposed to a graph based parser; we consider lexical preference exclusively for PP-attachment and test multiple levels of lexical specification; and our preference scores are calculated using both PMI and a novel neighborhood-based metric.

5.1.2 PP-Attachment Preference Types

In the lexical preference experiments for this chapter, we focus on the problem of modeling PP-attachment preference. The difficulty for dependency parsers to correctly attach prepositions to their correct governors, due to large amounts of artificial syntactic ambiguity, can potentially be mitigated by a direct modeling of lexical syntactic and semantic preference.

It is important to note that there needs to be a strong lexical component in order for this type of syntactic and semantic preference modeling to succeed. For PP-attachment, a strong lexical component is certainly present. Verb lexemes in both English and French are known to subcategorize for indirect object dependent

PPs, as in the case of the French verb 'rêver' ("to dream") and the subcategorized indirect object PP headed by the preposition 'de' ('of'). Additionally, it is also clear that the lexical semantics between governors (primarily nouns and verbs) and dependent PPs plays a role in the acceptability or preference for certain attachments over others. For example, consider the attachment of a PP headed by 'avec' ('with') as a modifier of either the verb 'manger' ("to eat") or the noun 'salade' ('salad'): In the sentence "J'ai mangé une salade avec des champignons" ("I ate a salad with mushrooms"), lexical semantics favors the attachment of the PP to the noun. On the other hand, in the sentence "J'ai mangé une salade avec une fourchette" ("I ate a salad with a fork") the attachment of the PP to the verb is semantically favored.

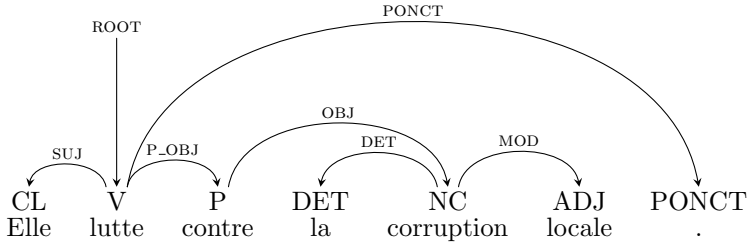
In contrast, for coordination, another difficult attachment type that we initially thought of modeling with similar techniques, it is unclear to what extent a localized lexical preference model could help with disambiguating syntactic structure. Coordination is a higher-level process that is not subcategorized for, and while semantic knowledge is certainly needed to disambiguate its scope, our intuition is that our local lexical preference modeling will not help much. Related works seem to support this hunch, with the results reported by Mirroshandel et al. (2012) indicating that lexical affinity scores computed for coordination do not improve parsing accuracy.

Our goal is then to model the level of attachment preference between potential governor-dependent pairs in cases where the dependent is a PP. The notion of preference is intentionally left vague, as different types of syntactic and semantic requirement can be captured implicitly through frequencies of observed dependencies in a large automatically-parsed corpus. We can nonetheless try to focus on different types of requirement by varying the levels of lexical specificity of the dependent PP position. The below sections identify two standard levels that we will consider in our experiments.

Subcategorization Preference

The first level of lexical specificity that we consider for the dependent PP is one that approximately matches the notion of subcategorization. We base our modeling on the traditional notion of subcategorization that is applicable to both English and French, with our guide on French verb subcategorization being the manually-developed resource Dicovalence (Eynde and Mertens, 2003; Mertens, 2010); in our experiments, we ended up using it as an alternate lexical preference resource to compare against our semi-supervised approach.

Although subcategorization is intrinsically associated with verb governors, deverbal nouns can also serve as predicates with similar properties. Consider the pair



Subcategorization	Lexical Association
(V, 'lutter'), (P, 'contre', N)	(V, 'lutter'), (P, 'contre', N, 'corruption')

Figure 5.1: Identification of governor and dependent PP tuples at two levels of lexical specificity for PP-attachment lexical preference in the sentence: "Elle lutte contre la corruption locale." ("She fights against local corruption.")

of verb 'lutter' ("to fight") and noun 'lutte' ('fight'): both subcategorize for a PP headed by one of the prepositions 'pour' ('for') or 'contre' ('against'). Given this fact, as well as our desire to cast a wider net rather than be too restrictive in our statistical modeling of preference, we consider subcategorization preference involving both verb and noun governors. We always include the lemma of the governor in our modeling, as subcategorization requirements and preferences are lexicalized.

As for the dependent PP, we follow the general formulation of PP roles in subcategorization frames by including the lemma of the head preposition as well as the POS category corresponding to its object. The POS of the PP's object is necessary to include because it distinguishes between the two major distributions of PPs with nominal objects and with verbal objects. While a sentential object is also possible, as in "avant que tu sois arrivé" ("before [that] you had arrived"), most prepositions followed by 'que' in this way are annotated in the FTB as compound word complementizers like 'avant_que', and are thus not pertinent to our modeling.[1]

As an example of an instance of subcategorization in a sentence, consider "Elle lutte contre la corruption locale" ("She fights against local corruption"). We can observe here the subcategorization pair with governor tuple of coarse-grained POS and lemma (V, 'lutter') and dependent tuple of preposition coarse-grained POS and lemma and object coarse-grained POS (P, 'contre', N). This example of subcategorization preference is represented on the left side of Figure 5.1.

[1]The constructions "à ce que" and "de ce que" ("of that which") are not annotated as compounds, but due to their rarity we ignore them in our modeling.

Lexical Association Preference

For a richer level of lexical specificity, we use what we have provisionally termed as *lexical association* preference. This is similar to subcategorization preference, except that we additionally use the lemma of the PP's object. As noted earlier, subcategorization requirements are strictly syntactic in nature, yet the resolution of ambiguity for certain linguistic constructions like PP-attachment often requires recourse to semantics. While lexical association is therefore an attractive option to consider, we nonetheless note that using a higher level of lexical specificity has the downside of increasing the amount of data sparseness encountered when computing lexical preference scores, with less reliable estimates of lexical preference pair frequencies.

Continuing with our previous example for subcategorization, we can see that for the sentence "Elle lutte contre la corruption locale" ("She fights against local corruption"), the lexical association pair will have a governor tuple of coarse-grained POS and lemma (V, 'lutter') and a dependent tuple of preposition coarse-grained POS and lemma and object coarse-grained POS and lemma (P, 'contre', N, 'corruption'). This is represented on the right side of Figure 5.1.

5.1.3 Statistical Preference Metrics

Our statistical metrics for the extraction of lexical preference scores from a large parsed corpus are based on the same basic concepts that we previously introduced in Chapter 4. Our terminology and methods remain based on the work of Lin (1998), which used word context relations to calculate distributional lexical similarity, and the subsequent work of Curran (2004), which distinguished between weight and measure functions and evaluated different functions on a semantic similarity task for English. In essence, we follow those methods but stop after the weighting of context relations, as we do not require the calculation of similarity between like lexical terms.

The basic framework we use for statistical methods for lexical preference is thus as follows:

1. Extraction: obtain counts of governors and PP contexts in which they appears from a parsed text corpus.

2. Weight: modify the raw counts in order to compute a score that better reflects the preference between the governor and the PP.

The governors and PP contexts are as we have just finished describing above, with lexicalized governors being restricted to nouns and verbs, and dependent PPs

restricted to those with nouns and verbs in the PP's object position, at two different levels of lexical specificity. The two methods for weighting raw counts into preference scores between a governor and a dependent PP are described below.

A Traditional Metric: PMI

Our first weight metric is the widely-used PMI metric, which we found to be the most effective among a number of different metrics when building distributional thesauri in Chapter 4, and which has been the metric of choice in most related work on using calculated lexical preference scores to improve parsing. The formula for PMI appeared in Table 4.2.

A Novel Neighborhood-Based Metric: NRF

Our second weight metric uses a novel neighborhood-based approach, combining the basic notion of relative frequency with the useful information that can be gained by looking at plausible alternative governors in a syntactic neighborhood around a dependent PP. The typical metric for relative frequency, the formula of which appears in Table 4.2 from the previous chapter, is not a very robust context relation weight metric in distributional lexical semantics, as evidenced in previous experiments such as that of Curran (2004) or in our earlier intrinsic distributional thesaurus evaluation. However, the idea of using a metric conditioned on the presence of a dependent PP, rather than a symmetric metric like PMI or others, is attractive: since we will be using preference scores as features for neighborhood correction classifiers, our problem setting has a dependent PP as a given and the choice of governor as essentially the variable to be predicted.

However, we can go further than simply trying to model a conditional probability of the form $p(gov|PP)$, which would correspond to an artificial problem in which the choice of a governor for a dependent PP would take place given no additional information, as if any governor could potentially be generated. We know, rather, that by construction the candidate governors we consider are those in the syntactic neighborhood surrounding the PP and its predicted governor from the first-stage transition-based parser output. Therefore, in defining our metric we should additionally condition on the presence of a potential governor appearing in the neighborhood of the dependent PP, since if it does not appear in the neighborhood then it cannot possibly be chosen as the new governor for the PP.

We start with the simply probability of generating a governor g given a dependent d, the latter of which is a PP including both the preposition and its object. To estimate this probability we use a basic relative frequency estimation, where $f(g, d)$

is the frequency of a pair of governor g and dependent, d and $*$ indicates a sum over all possible values in a given position. The probability and its estimation are:

$$\hat{p}(g|d), \quad \frac{f(g,d)}{f(*,d)} \tag{5.1}$$

Our proposal is then to consider the neighborhood-based probability of selecting a governor g given both a dependent d and the information that g is in the neighborhood N_d of candidate governors surrounding d. To estimate this probability we use a *neighborhood-based relative frequency (NRF)* estimation, where the new term $f(g \in N_d)$ is the frequency of g appearing in a candidate neighborhood of d. The probability and its estimation are:

$$\hat{p}(g|d, g \in N_d), \quad \frac{f(g,d)}{f(g \in N_d)} \tag{5.2}$$

In comparing these two probability estimation metrics, we note that basic relative frequency has an unwanted tendency to place an overly high weight on governors that simply appear frequently in the corpus, while NRF corrects for this tendency by restricting the counts for a possible governor to cases in which it appears in the neighborhood for a dependent. Consider the very frequent verb 'être' ("to be"), which subcategorizes for a number of PPs, including the one containing the preposition 'contre' ("against") and a noun PP object. It will receive a much higher basic relative frequency with respect to that PP compared to other verbs that are just as good if not better fits, such as the verb 'lutter' ("to fight"), simply because 'être' is so frequent. By taking into account the frequency of appearance in the PP's syntactic neighborhood in the denominator as opposed to just the PP's frequency, the NRF metric can give a better accounting of which governor our correction model should actually be selecting. It could discover, for instance, that in a low proportion of cases where 'être' appears in the neighborhood of that PP 'être' is actually the predicted governor, while in a higher proportion of cases where 'lutter' appears in the neighborhood of that PP 'lutter' is the predicted governor.

In order to carry out this new type of estimation over an automatically-parsed corpus, the only work beyond our usual methods is to identify the syntactic neighborhoods of dependent PPs and keep track of some additional frequencies.

5.2 PP-Attachment Preference Experiments

In this section we describe our final set of experiments related to lexical preference as a way to improve the accuracy of PP-attachments in dependency parsing. We use as

our base method a two-stage transition-based parsing and neighborhood correction system as presented in Chapters 2 and 3, respectively. We only consider the use of lexical preference resources for correction, due to the fact that the composition of a PP is not fully determined prior to its attachment during first-stage transition-based parsing. Within the correction models, we compare the integration of two different lexical preference resources: the first consists of lexical preference scores adapted from the hand-built Dicovalence verb subcategorization resource, while the second is built using statistical metrics over a large automatically-parsed journalistic corpus and results in our two-stage parser becoming a semi-supervised system.

5.2.1 Methods and Setup

We now present the setup of our semi-supervised lexical preference experiments. For all methodological details concerning parsing with ARC-EAGER and neighborhood correction, see our previous descriptions in Section 2.3 and Section 3.3, respectively. As was the case for our previous experiments, we used our own Python implementation of the algorithms and methods from this chapter; we reiterate that we plan to release a package of the code used in this thesis following its publication. Our methodology for experiments in lexical preference involves the following elements: building a lexical preference resource from an automatically-parsed and corrected external corpus; preprocessing the Dicovalence for use as a lexical preference resource along the same lines as our semi-supervised one; and finally defining novel features for lexical preference in our neighborhood correction models.

Calculation of Lexical Preferences

For the calculation of lexical preferences for PP-attachment, we considered a number of different settings. One variable was the type of preference to obtain, with the two options being subcategorization (less lexically specified) and lexical association (more lexically specified). A second variable was the statistical metric to use in our calculations, with the two options being PMI and our novel neighborhood relative frequency metric NRF. Finally, in order to make the resource construction process tractable, as well as to account for noise due to automatic parsing errors, we implemented a minimum frequency cutoff for both governors and dependent PPs; a variety of cutoffs were tested, and a preliminary evaluation of different cutoff values on the FTBDep development set will be presented in the results section.

As noted earlier in Section 5.1.2, governors were restricted to those with POS category of verb or noun, and the dependent PPs were restricted to those with PP object POS category of verb or noun. An additional step we took to account for noise

in the automatically tokenized and POS-tagged corpus was to define a set of 'valid' preposition lemmas based on those appearing in the FTBDep development set, so as to ignore word tokens wrongly tagged as prepositions in the external corpus.

The unlabelled French corpus we considered as the basis for computing our lexical preference scores was the AFP, which we used in Chapters 3 and 4 for self-training and lexical generalization experiments, respectively. As before, the corpus was preprocessed using the Bonsai tool, which performed sentence segmentation, word tokenization, and additionally, consistent with our automatic annotation of the FTBDep in Section 2.3, performed automatic POS tagging with the MElt package and lemmatization with the Lefff lexicon. Additionally, the corpus was parsed and corrected using a two-stage parsing system with ARC-EAGER transition-based parsing and neighborhood correction, as first presented in Chapter 3.

Preprocessing Dicovalence

The next step in our experimental setup was to convert the Dicovalence resource into a format that directly listed subcategorization preference scores between governors and dependent PPs. For these experiments, we used version 2.0 of Dicovalence (Mertens, 2010). The Dicovalence resource is structured so that each verb subcategorization frame is granted a separate multi-line entry, with a list of argument paradigms followed by a list of POS categories that are accepted in those positions, with notably preposition lexemes explicitly listed for those paradigms corresponding to PP-attachment. From these entries we extracted the desired PP lexical preference information; an example Dicovalence entry and subsequent extraction are shown in Figure 5.2. Once all relevant subcategorization instances were identified, we simply assigned a score of 1 to the attested pairings of governor and dependent PP, with an implicit score of 0 given to any pairing not found in the resource.

Definition of Lexical Preference Features

The final part of our experimental setup was the definition of novel features for lexical preference in our neighborhood correction models. Table 5.1 lists the features templates for the oracle of neighborhood parse correction, with a description of most of these features already discussed for our correction experiments in Section 3.3.

The new feature templates are SUBCAT_{c,d,d_o} and $\text{LEXASSOC}_{c,d,d_0}$, encoding subcategorization and lexical association preference, respectively. These feature templates are different from the others in that they are real-valued rather than categorical; while categorical feature templates are represented using separate indicator features each encoding one categorical value, each lexical preference feature template

Dicovalence Entry

VAL	abandonner: P0 P1 P2
VTYPE	predicator simple
VERB	ABANDONNER/abandonner
NUM	60
EG	mon oncle a abandonné tous ses biens à sa maîtresse
TR_DU	overleveren (aan), prijsgeven (aan)
TR_EN	abandon, bequeath, demise (to)
FRAME	subj:pron|n:[hum], obj:pron|n:[nhum,?abs], objà:pron|n:[hum]
P0	qui, je, nous, elle, il, ils, on, celui-ci, ceux-ci
P1	que, la, le, les, en Q, ça, ceci, celui-ci, ceux-ci
P2	qui, me, lui, leur, celui-ci, ceux-ci
RP	passif être, se passif
AUX	avoir

Subcategorization

(V, 'abandonner'), (P, 'à', N)

Figure 5.2: Identification of a PP subcategorization preference from an entry in Dicovalence for the verb 'abandonner' ("to abandon"). The subcategorized PP consists of the preposition 'à' ('to') with a noun PP object, with this information being gleaned from the presence of objà in the FRAME field.

is represented directly with a single real-valued feature. In order to avoid negatively impacting the SVM learning process, we required that values remain in the range $[-1, 1]$. Since NRF is a probabilistic metric, it naturally resides in this range. For PMI, we used a simple normalization technique where the highest positive score observed during calculation was stored, and then each positive score was divided by the highest score.[1]

By using separate features for these two types of preference, we were able to experiment with learning models that used only one type of information or both types simultaneously. We also note that this feature template representation remains agnostic to the type of dependency between a candidate c and a dependent d. However, since our preference scores have been calculated exclusively for dependent PPs these features were effectively restricted to PP-attachment preference in our experiments.

[1] A quirk of methods for estimating joint lexical probabilities from corpora is that pairs appearing together at least once rarely have a negative PMI, while pairs that never appear together have a PMI of $-\infty$. We simply mapped all negative PMI to a feature value of -1.

Correction Templates	
Simple:	POS_d
	LEM_d
	OPOS_d
	POS_c
	LEM_c
	HPOS_c
	ISPR_c
Derived:	NDEPS_c
	$\text{DIST}_{c,d}$
	$\text{DIR}_{c,d}$
	$\text{PATH}_{c,d}$
	$\text{PUNC}_{c,d}$
	$\text{LPOS}_{c,d}$
	$\text{LLAB}_{c,d}$
	$\text{RPOS}_{c,d}$
	$\text{RLAB}_{c,d}$
Preference:	SUBCAT_{c,d,d_o}
	$\text{LEXASSOC}_{c,d,d_o}$

Table 5.1: Feature templates for neighborhood parser correction, both simple ones over individual word forms and derived ones that use multiple words or surrounding syntactic structure. Novel lexical preference features apply to preposition dependents only.

5.2.2 Results

This section describes the results of our lexical preference experiments, where we used neighborhood correction as the base approach over which novel lexical preference features for PP-attachment were evaluated.

Preliminary Development Evaluation

We first evaluated a number of of different settings on the development set, with preposition UAS reported in Table 5.2. The first setting corresponds to the Dicovalence approach, which uses that hand-built resource for French as a source of 0-1 verb subcategorization features in our neighborhood correction models. For this setting, we did not have to tune any parameters or consider different approaches, as the 0-1 PP-attachment preference scores were converted directly from the Dicovalence subcategorization frames, and only the $\text{SUBCAT}_{c,d}$ feature template was introduced to the neighborhood correction model. This approach resulted in a minor improvement over the baseline two-stage parsing and correction system, so we decided to carry it forward to our final evaluation on the FTBDep test set.

The next two groups of settings correspond to our main approach, which uses features derived from semi-supervised lexical preference scores calculated from an

	Settings	UAS
	ARC-EAGER	83.5
Baseline	+CORRECTION	83.8
Dicovalence	SUBCAT	84.1
PMI	SUBCAT	
	k_d=50, k_g=50	84.4
	k_d=100, k_g=100	84.3
	k_d=200, k_g=200	84.5†
	k_d=500, k_g=500	84.4
	LEXASSOC	
	k_d=50, k_g=50	84.5
	k_d=100, k_g=100	84.5†
	k_d=200, k_g=200	84.5
	k_d=500, k_g=500	84.3
	SUBCAT+LEXASSOC	84.5*
NRF	SUBCAT	
	k_d=100, k_g=5	84.5
	k_d=100, k_g=10	84.3
	k_d=200, k_g=5	84.6
	k_d=200, k_g=10	84.5
	k_d=500, k_g=5	84.6†
	k_d=500, k_g=10	84.5
	LEXASSOC	
	k_d=100, k_g=5	84.4
	k_d=100, k_g=10	84.6†
	k_d=100, k_g=20	84.4
	k_d=200, k_g=5	84.4
	k_d=200, k_g=10	84.5
	k_d=200, k_g=20	84.6
	SUBCAT+LEXASSOC	84.9*

Table 5.2: Preposition UAS results, in percent, over the FTBDep development set with different lexical preference settings for correction. The baseline uses no lexical preference features, the Dicovalence approach uses SUBCAT only, and the PMI and NRF approaches also use LEXASSOC as well as varying frequency cutoffs for dependents (k_d) and governors (k_g). † indicates a best setting later combined into a SUBCAT+LEXASSOC setting (last rows). $*$ indicates a statistically significant improvement for SUBCAT+LEXASSOC over the baseline.

external automatically-parsed and corrected French corpus. The first group represents the traditional PMI metric for score calculation, and within that group we tested four minimum frequency cutoffs (50, 100, 200, 500) for the governors (k_g) and dependent PPs (k_d), with the same cutoff applied to both. The second group represents our novel NRF metric for score calculation, and within that group we tested three cutoff values for dependent PP frequency k_d (100, 200, 500) and three cutoff values for governor frequency k_g relative to a particular dependent PP (5, 10, 20); for NRF, governor frequency was calculated separately for each dependent PP,

Settings	PPs	Govs	Pairs
Dicovalence: SUBCAT	46	2k	3k
PMI: SUBCAT, k_d=200, k_g=200	159	5k	89k
PMI: LEXASSOC, k_d=100,k_g=100	18k	7k	2m
NRF: SUBCAT, k_d=500, k_g=5	147	32k	161k
NRF: LEXASSOC, k_d=100, k_g=10	18k	14k	479k

Table 5.3: Size of lexical preference resources for Dicovalence, and for each combination of statistical metric (PMI or NRF) and lexical preference type (SUBCAT or LEXASSOC) using optimal minimum frequency cutoff values over the AFP corpus. Lists unique number of PPs, governors, and attested dependency pairs.

and counted every time the governor appeared in the candidate neighborhood of that PP. An additional grouping parameter for the semi-supervised approaches was the use of subcategorization or lexical association preference.

For the semi-supervised lexical preference approaches, we identified the best development minimum frequency cutoff values for each pair of statistical metric (PMI or NRF) and preference type (subcategorization and lexical association), and then evaluated a best setting for each statistical metric that simultaneously used both preference feature types. These best settings are also shown in Table 5.2. We can already see that they appear to be better than the Dicovalence approach, with the preposition UAS being higher for PMI (84.5) and for NRF (84.9) compared to Dicovalence (84.1).

To give a sense of the size of our final lexical preference resources, Table 5.3 lists the size of resources for Dicovalence and each set of optimal minimum frequency cutoff values for the best semi-supervised approaches. We can see that Dicovalence covers a lower number of unique PPs, governors, and dependency pairs compared to the semi-supervised approaches. We also note that lexical association has a much higher number of unique PPs and dependency pairs compared to subcategorization, which is a result of taking into account the lemma of the PP's object.

Final Test Evaluation

For our final evaluation over the FTBDep test set, we considered only a single *best* setting for each of the three approaches (Dicovalence, PMI, NRF) as determined by our development set evaluation. While for simplicity preposition UAS was used to find the best settings in the development set evaluation, we now report full overall LAS and UAS in addition to preposition LAS and UAS in Table 5.4.[1]

[1] We do not report statistical significance for overall LAS and UAS, as it would be misleading. Only the corrective model for prepositions changes between the baseline and the new approaches, meaning that other dependents are almost all attached to same governors regardless of the approach and should not factor into the statistical test.

	Settings	Overall		Preps	
		LAS	UAS	LAS	UAS
	ARC-EAGER	87.1	89.7	78.3	84.0
Baseline	+CORRECTION	87.5	90.2	78.8	84.6
Dicovalence	SUBCAT	87.5	90.2	78.7	84.4
PMI	SUBCAT+LEXASSOC	87.5	90.2	79.0	84.9
NRF	SUBCAT+LEXASSOC	87.6	90.3	79.4*	85.2*

Table 5.4: Overall and preposition LAS and UAS results, in percent, over the FTBDep test set when using different lexical preference settings for neighborhood correction. The baseline uses no lexical preference features, the Dicovalence approach uses subcategorization (SUBCAT) features only, and the PMI and NRF approaches use a combination of subcategorization and lexical preference (LEXASSOC) features. $*$ indicates a statistically significant improvement over the baseline for preposition LAS or UAS.

We first note that using 0-1 verb subcategorization scores from Dicovalence actually led to a slight decrease in preposition LAS and UAS, which indicates that this approach is not well-suited to the problem of improving PP-attachment performance. This can be explained in part by certain characteristics of Dicovalence: it has rather low coverage, and it includes rare senses for frequent verbs that may actually mislead the correction model. This result seems to support our motivating hypothesis presented at the beginning of this chapter: in order to deal with difficult cases of ambiguity in linguistic constructions like PP-attachment, one needs to go beyond strict syntactic requirements. As a hand-built subcategorization resource, Dicovalence doesn't capture nuanced subcategorization preferences or more lexically specified preferences that would include the object of the PP. As we noted then, statistical approaches for parsing are already set up to capture syntactic requirements indirectly through the syntactically annotated training corpus used, so it is possible that strict subcategorization features may have been redundant to a large extent.

We now move on to look at the parsing improvements achieved when using the semi-supervised approaches to integrating lexical preference scores. Both subcategorization (SUBCAT) and lexical association (LEXASSOC) were used jointly, in order to capture both syntactic and more lexically specified semantic preference and requirements. Of the two statistical metrics, NRF was the one that obtained the best overall parsing improvement, with a statistically significant improvement in preposition LAS and UAS compared to the baseline two-stage parsing and correction system. PMI obtained a smaller, not statistically significant improvement in preposition LAS and UAS over the baseline. Comparing NRF and PMI, however, the difference was unfortunately not statistically significant.

Despite the fact that NRF was not statistically better than PMI, it is nonetheless the case that NRF was the best performing approach and was also the only

lexical preference setting that was statistically better than the baseline. This is an interesting result, as PMI is the standard statistical metric used in the literature to compute scores for lexical preference from a corpus, and as described in Chapter 4 it is also to our knowledge the most commonly used metric for weighting context relations in distributional lexical semantics methods at large. However, we are not surprised that NRF performs better here, as it is adapted to our particular task through its use of candidate governor neighborhoods.

The use of syntactic neighborhood information in a statistical metric is thus a promising idea, achieving modest parsing improvements for PP-attachment but having potential applications to other tasks as well. As we noted when initially describing the NRF metric, it is likely too much of an abstraction to think about governor-dependent pairs as being generated in a vacuum. By restricting the frequencies we consider to cases in which a potential governor and dependent appear in the same local syntactic context, and only then counting the instances in which they are attached or not, we end up with a more informed metric for how likely they are to be attached when observed again in that context. It would be interesting to see how this type of metric could be applied to tasks that use distributional lexical semantics, such as the creation of distributional thesauri (c.f. Chapter 4) or the problems of automatic word sense disambiguation and acquisition of semantic classes (c.f. Section 5.1.1).

Conclusion

In this thesis we explored ways to improve efficient statistical dependency parsing, with French as our language of application and the French Treebank converted to dependencies (FTBDep) as our primary source of data. We tested a number of different methods, with our first research thread focused on algorithmic and syntactic feature representation issues, and our second research thread focused on semi-supervised methods over unlabeled corpora.

Syntactic Context in Parsing and Correction

For the first main thread, we worked within the transition-based parsing and neighborhood correction algorithms to introduce additional syntactic context into attachment decisions. This aspect of our research was motivated by the fact that transition-based dependency parsing algorithms attain high computational efficiency, relative to more complex approaches used for phrase-structure parsing or graph-based dependency parsing, by making greedy locally-optimal attachment decisions. In Chapter 2 we introduced a variant ARC-EAGER-MC transition system, based on the well-studied ARC-EAGER transition system. Our goal was to introduce additional syntactic context into attachment decisions by considering multiple candidate governors simultaneously for right-directed dependencies. While this variant had the downside of a worst-case quadratic time parsing complexity, compared to the linear time parsing complexity of ARC-EAGER, it lead to a small improvement in preposition UAS that suggested a multiple-candidate approach is useful for improving highly ambiguous right-directed dependencies that include the linguistic phenomenon of PP-attachment.

This result led us to our investigation in Chapter 3 into parse correction modeling, specifically an existing linear-time neighborhood correction algorithm that by

design revises each dependent by considering multiple candidate governors from the syntactic neighborhood surrounding its predicted governor from the initial parse tree. A small innovation on our part was to make explicit the ranking aspect of these revisions; we moved away from the original formulation of the learning problem, where the model is given independent binary examples for each candidate, and instead used an explicit ranking formulation of the learning problem with examples created from pairwise ranking constraints between the correct candidate governor and each incorrect candidate governor. We also expanded the set of features to include the dependent's object (in the case of PP-attachment) or right conjunct (in the case of coordination), which from a linguistic standpoint are important for determining the correct governor; note that these features are not available during transition-based ARC-EAGER parsing. We found that a two-stage system using a second-stage neighborhood corrector obtained statistically significant improvements in LAS and UAS, overall and specifically for PP-attachment and coordination, when compared to first-stage ARC-STANDARD or ARC-EAGER parsing alone.

Semi-Supervised Lexical Methods

In the second main thread of our thesis, we used the resulting efficient two-stage parser as a basis from which we could look outward, investigating semi-supervised methods over unlabeled corpora to tackle the problems of data sparseness and lexical class and preference modeling. In the latter part of Chapter 3 we tested a standard self-training approach that parses an unlabeled external corpus and incorporates some of those parses back into the training set. We tried this both within the journalistic domain of the FTBDep and for adaptation to a medical domain, in both cases unfortunately obtaining no improvement in parsing over the baseline two-stage parser.

We then turned to more focused semi-supervised approaches concerning lexical modeling. In Chapter 4 we used an automatically parsed corpus to build distributional thesauri, which were then integral to the construction of three spaces of generalized lexical classes: lemmas ranked by distributional similarity, distributional word clusters, and word senses automatically ranked based on distributional similarity. Our contribution was to use a multiple assignation strategy for replacing word forms with lexical classes in features: a feature template normally represents a categorical variable with each value assigned an indicator feature, and in the case of multiple assignation we allowed lexical categorical variables to take on multiple values simultaneously. For instance, when replacing the word form 'avocat' with multiple semantic senses, both indicator features $\text{SENSE}_{\beta[0]}$='avocado' and $\text{SENSE}_{\beta[0]}$='lawyer' would fire when 'avocat' is encountered. We obtained statistically significant im-

provements in LAS and UAS for two-stage parsing when replacing a word form with two lemmas — the lemmatized form and its most distributionally similar lemma — for parsing both in-domain and when adapted to the medical domain, with the latter result obtained when using an external 'bridge' corpus containing both journalistic and medical text.

Finally, in Chapter 5 we used an automatically parsed corpus to extract lexical preference scores for PP-attachment. Given that our two-stage parser has access to the key information required for disambiguating PP-attachment — the candidate governor, the preposition, and the PP's object — we were able to directly introduce lexical preference scores as features into the neighborhood correction model. The methods we used to obtain the scores were mostly standard, with one type of preference corresponding to a less lexically-specified correlate of subcategorization and the other type of preference modeling a more lexically-specified association. Our contribution was to test an alternate preference metric to the standard point-wise mutual information (PMI) widely used in the literature; we introduced neighborhood-based relative frequency (NRF), which estimates the probability of a candidate governing a dependent PP given that it appears in the syntactic neighborhood of that PP. NRF addressed the problem with basic relative frequency of favoring governors that simply appear very often in the corpus, and because it uses neighborhoods as the basis of its calculation it also has the advantage of producing scores that are especially adapted to the problem of neighborhood correction. In our experiments, we first tested the use of 0-1 lexical preference scores extracted from a verb subcategorization resource for French, finding no improvement in PP-attachment for the two-stage parser. For the semi-supervised settings, we found that using preference scores for both levels of lexical specificity worked better than using either alone, and only the NRF metric led to a statistically significant improvement in preposition LAS and UAS for the two-stage parser, though the difference between and NRF and PMI was not statistically significant.

Looking Forward

Our results are on the one hand encouraging, as a number of the approaches we tested led to statistically significant improvements in dependency parsing, but on the other hand discouraging as the improvements were modest in most cases. There remain a number of possible paths for future research, particularly as pertains to semi-supervised parsing with lexical resources: soft clustering methods would seem to be useful in conjunction with the multiple assignation strategy for feature replacement of lexical classes; more sophisticated lexical classes could be obtained using LDA (Ó Séaghdha, 2010; Ritter et al., 2010); and in-context disambiguation

of word sense (Brody and Lapata, 2008; Erk and Padó, 2008; Erk and Pado, 2010), and consequently of selectional preference, could make the use of lexical resources more effective.

Looking at the current state of dependency parsing at large, we note that in just the past few years a wealth of new research has cropped up. The work of Goldberg and Elhadad (2010) provides an interesting compromise approach between the two major families of dependency parsers, which are the globally-optimized cubic-time graph-based parsers and the locally-optimized linear-time transition-based parsers. It uses what the authors term an easy-first approach to making attachments, building dependencies in a bottom up fashion, and interestingly, its $O(n\log n)$ computational complexity lies between those of the two major families of dependency parsers. Other very recent works have exploited the use of a generalized perceptron (Collins, 2002) in combination with beam search, as first proposed for transition-based parsing in the work of Zhang and Clark (2008). Beam search allows for up to k transition sequences to be stored at any given point during parsing, with the generalized perceptron fulfilling the need for a global score comparable across transition sequences. (Zhang and Nivre, 2011) add to this framework syntactically rich feature templates for ARC-EAGER parsing that increase accuracy substantially, while (Bohnet, 2011) uses a fast hash kernel with passive-aggressive updates that takes into account 'negative' features, or those that would not be encountered in gold transition sequences during training. Both works obtain results for English that are surprisingly on par with the best graph-based parsers, possibly settling the dispute between the two parsing families for good.

Given the existence of these new dependency parsers, it would be interesting to see whether neighborhood parse correction and automatically-built lexical resources could still be beneficial. We note that transition-based parsing remains an approach that uses incomplete syntactic context during attachment decisions, as the parse tree is built incrementally; neighborhood parse correction, with access to complete surrounding syntactic context for attachment decisions, might then provide an effective second pass even for highly-accurate transition-based parsers that use beam search. We also note that the issues of data sparseness and generalization to different domains are tied to the source treebank and not to the choice of parsing algorithm; semi-supervised lexical methods using distributional lexical classes or lexical preference scores might then improve both in-domain and out-of-domain parsing accuracy even for the new crop of parsers.

Bibliography

A. Abeillé and N. Barrier. Enriching a French treebank. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, 2004. 32

A. Abeillé, L. Clément, and F. Toussenel. Building a treebank for French. *Treebanks*, pages 165–187, 2003. 29, 30, 31, 32, 33, 34, 143

E. Agirre and D. Martinez. Learning class-to-class selectional preferences. In *Proceedings of the 2001 Workshop on Computational Natural Language Learning*, pages 15–22, 2001. 144

E. Agirre, T. Baldwin, and D. Martinez. Improving parsing and PP attachment performance with sense information. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 317–325, 2008. 120, 140

E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, and A. Soroa. A study on similarity and relatedness using distributional and WordNet-based approaches. In *Proceedings of the 2009 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27, 2009. 121

E. Agirre, K. Bengoetxea, K. Gojenola, and J. Nivre. Improving dependency parsing with semantic classes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 699–703, 2011. 120, 128

A. Aho, J. Ullman, and R. Sethi. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, 1986. 57, 61

A. Arun and F. Keller. Lexicalization in crosslinguistic probabilistic parsing: The case of French. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 306–313, 2005. 38

G. Attardi. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 166–170, 2006. 107

G. Attardi and M. Ciaramita. Tree revision learning for dependency parsing. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 388–395, 2007. 2, 89, 92, 107

G. Attardi and F. Dell'Orletta. Reverse revision and linear tree combination for dependency parsing. In *Proceedings of the 2009 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 261–264, 2009. 90

M. Atterer and H. Schütze. Prepositional phrase attachment without oracles. *Computational Linguistics*, 33(4):469–476, 2007. 91

M. Bacchiani, M. Riley, B. Roark, and R. Sproat. Map adaptation of stochastic grammars. *Computer speech & language*, 20(1):41–68, 2006. 99

J.M. Balfourier, P. Blache, M.L. Guénot, and T. VanRullen. Comparaison de trois analyseurs symboliques pour une tâche d'annotation syntaxique. In *Actes de la 12ème conférence sur le traitement automatique des langues naturelles*, pages 41–48, 2005. 50

A.L. Berger, V.J.D. Pietra, and S.A.D. Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996. 41

S. Bergsma, D. Lin, and R. Goebel. Discriminative learning of selectional preference from unlabeled text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 59–68, 2008. 145

D.M. Bikel. A statistical model for parsing and word-sense disambiguation. In *Proceedings of the EMNLP/VLC-2000*, pages 155–163, 2000. 119

S. Bird, E. Loper, and E. Klein. *Natural Language Processing with Python*. O'Reilly Media Inc., 2009. 81, 133

J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128, 2006. 98

L. Bloomfield. *Language*. Holt, New York, 1933. 9

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. The Prague dependency treebank. In *Treebanks*, pages 103–127. Springer, 2003. 29

B. Bohnet. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 89–97, 2010. 146

B. Bohnet. Comparing advanced graph-based and transition-based dependency parsers. In *Proceedings of the International Conference on Dependency Linguistics*, pages 282–289, 2011. 163

B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992. 43

D. Bourigault. Upery: un outil d'analyse distributionnelle étendue pour la construction d'ontologies à partir de corpus. In *Actes de la 9ème conférence sur le traitement automatique des langues naturelles*, pages 75–84, 2002. 119

J. Bresnan, A. Cueni, T. Nikitina, and H. Baayen. Cognitive foundations of interpretation. In G. Boume, I. Kraemer, and J. Zwarts, editors, *Predicting the dative alternation*, pages 69–94. Amsterdam : Royal Netherlands Academy of Science, 2007. 14

T. Briscoe and J. Carroll. Automatic extraction of subcategorization from corpora. In *Proceedings of the fifth conference on Applied natural language processing*, pages 356–363, 1997. 145

S. Brody and M. Lapata. Good neighbors make good senses: Exploiting distributional similarity for unsupervised WSD. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 65–72, 2008. 163

P.F. Brown, P.V. Desouza, R.L. Mercer, V.J.D. Pietra, and J.C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992. 120

S. Buchholz and E. Marsi. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164, 2006. 49, 52, 55, 56

R.C. Bunescu. Learning with probabilistic features for improved pipeline models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 670–679, 2008. 134

M. Candito and B. Crabbé. Improving generative statistical parsing with semi-supervised word clustering. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 138–141, 2009. 120, 125

M. Candito and D. Seddah. Le corpus Sequoia : Annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical. In *Actes de la 19ème conférence sur le traitement automatique des langues naturelles*, 2012a. 104, 134

M. Candito and D. Seddah. Effectively long-distance dependencies in French: annotation and parsing evaluation. In *The 11th International Workshop on Treebanks and Linguistic Theories*, 2012b. 24

M. Candito, B. Crabbé, and P. Denis. Statistical French dependency parsing: Treebank conversion and first results. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*, 2010a. 24, 29, 34, 37, 38, 40, 52

M. Candito, J. Nivre, P. Denis, and E. Henestroza Anguiano. Benchmarking of statistical dependency parsers for French. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 108–116, 2010b. 52, 53, 54, 74, 80, 120

M. Candito, E. Henestroza Anguiano, and D. Seddah. A word clustering approach to domain adaptation: Effective parsing of biomedical texts. In *Proceedings of the 12th International Conference on Parsing Technologies*, 2011. 99, 104, 120, 134, 135

X. Carreras. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 957–961, 2007. 53

J. Carroll, G. Minnen, and T. Briscoe. Can subcategorisation probabilities help a statistical parser? In *Proceedings of the 6th ACL/SIGDAT Workshop On Very Large Corpora*, pages 118–126, 1998. 145

D. Cer, M.C. de Marneffe, D. Jurafsky, and C.D. Manning. Parsing to Stanford dependencies: Trade-offs between speed and accuracy. In *Proceedings of LREC*, pages 1628–1632, 2010. 52

C.C. Chang and C.J. Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011. 44, 78, 103

E. Charniak. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 598–603, 1997. 98

E. Charniak. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 124–131, 2001. 107

E. Charniak and M. Johnson. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 173–180, 2005. 90

N. Chomsky. *Syntactic Structures*. Mouton, Hague, 1957. 14, 16, 18

G. Chrupala. Efficient induction of probabilistic word classes with LDA. *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 363–372, 2011. 119

J. Cocke and J.T. Schwartz. *Programming languages and their compilers: Preliminary notes*. Courant Institute of Mathematical Sciences, 1969. 28

M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1–8, 2002. 163

M. Collins. Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637, 2003. 49

M. Collins and T. Koo. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70, 2005. 90

M. Collins, L. Ramshaw, J. Hajič, and C. Tillmann. A statistical parser for Czech. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 505–512, 1999. 107

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. ii, iii, 41, 42, 43

M.A. Covington. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th annual ACM southeast conference*, pages 95–102, 2001. 59

B. Crabbé and M. Candito. Expériences d'analyse syntaxique statistique du français. In *Actes de la 15ème conférence annuelle sur le Traitement Automatique des Langues*, 2008. 32, 35, 75

K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *The Journal of Machine Learning Research*, 3:951–991, 2003. 41

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585, 2006. ISSN 1532-4435. 41

J.R. Curran. *From distributional to semantic similarity*. PhD thesis, University of Edinburgh, 2004. 115, 118, 119, 121, 122, 123, 149, 150

I. Dagan, S. Marcus, and S. Markovitch. Contextual word similarity and estimation from sparse data. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 164–171, 1993. 119

I. Dagan, F. Pereira, and L. Lee. Similarity-based estimation of word cooccurrence probabilities. In *Proceedings of the 32nd annual meeting of the Association for Computational Linguistics*, pages 272–278, 1994. 118

I. Dagan, L. Lee, F. Pereira, et al. Similarity-based methods for word sense disambiguation. In *Proceedings of the 35th annual meeting of the Association for Computational Linguistics*, volume 35, pages 56–63, 1997. 118

H. Daumé III, A. Kumar, and A. Saha. Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 53–59, 2010. 99

P. Denis and B. Sagot. Coupling an annotated corpus and a morphosyntactic lexicon for state-of-the-art POS tagging with less human effort. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation*, 2009. 53, 76

D. Dowty. Thematic proto-roles and argument selection. *Language*, pages 547–619, 1991. 15

J. Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970. 28

J.M. Eisner. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 340–345, 1996. 53

K. Erk. A simple, similarity-based model for selectional preferences. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 216–223, 2007. 144

K. Erk and S. Padó. A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 897–906, 2008. 163

K. Erk and S. Pado. Exemplar-based models for word meaning in context. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 92–97, 2010. 163

K. Van den Eynde and P. Mertens. La valence: l'approche pronominale et son application au lexique verbal. *Journal of French Language Studies*, 13(1):63–104, 2003. 147

R.E. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, and C.J. Lin. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, pages 1871–1874, 2008. 54

C. Fellbaum, editor. *WordNet: An electronic lexical database*. MIT Press, 1998. 118, 127

O. Ferret. Discovering word senses from a network of lexical cooccurrences. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 1326–1332, 2004. 119

O. Ferret. Testing semantic similarity measures for extracting synonyms from a corpus. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*, 2010. 119

H. Gaifman. Dependency systems and phrase-structure systems. *Information and control*, 8(3):304–337, 1965. 22

P. Gamallo, A. Agustini, and G.P. Lopes. Clustering syntactic positions with similar semantic requirements. *Computational Linguistics*, 31(1):107–146, 2005. 144

Y. Goldberg and M. Elhadad. An efficient algorithm for easy-first non-directional dependency parsing. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750, 2010. 163

G. Grefenstette. *Explorations in automatic thesaurus discovery.* Kluwer Academic Press, 1994. 118

M. Gross. Constructing lexicon-grammars. In B.T.S. Atkins and A. Zampolli, editors, *Computational approaches to the lexicon*, pages 213–263. Oxford University Press, 1994. 120

K. Hall and V. Novák. Corrective modeling for non-projective dependency parsing. In *Proceedings of the Ninth International Workshop on Parsing Technologies*, pages 42–52, 2005. ii, iii, 2, 88, 89, 91, 92, 93, 95, 96, 97, 102, 107

K. Hara, M. Shimbo, H. Okuma, and Y. Matsumoto. Coordinate structure analysis with global structural constraints and alignment-based local features. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 967–975, 2009. 91

Z. Harris. Distributional structure. *Word*, 10(23):146–162, 1954. 116

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning.* Springer-Verlag, 2nd edition, 2008. 125

D.G. Hays. Dependency theory: A formalism and some observations. *Language*, 40 (4):511–525, 1964. 22

E. Henestroza Anguiano and M. Candito. Parse correction with specialized models for difficult attachment types. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 2011. 88

E. Henestroza Anguiano and M. Candito. Probabilistic lexical generalization for French dependency parsing. In *Proceedings of the ACL Workshop SPSemMRL*, 2012. 115, 120, 134

E. Henestroza Anguiano and P. Denis. FreDist: Automatic construction of distributional thesauri for French. In *Actes de la 18ème conférence sur le traitement automatique des langues naturelles*, pages 119–124, 2011. 119, 131

D. Hindle. Noun classification from predicate-argument structures. In *Proceedings of the 28th annual meeting on Association for Computational Linguistics*, pages 268–275, 1990. 118

D. Hindle and M. Rooth. Structural ambiguity and lexical relations. *Computational linguistics*, 19(1):103–120, 1993. 91

D. Hogan. Coordinate noun phrase disambiguation in a generative parsing model. In *In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, page 680, 2007. 91

C.W. Hsu and C.J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002. 45

J.J. Jiang and D.W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *International Conference on Research in Computational Linguistics*, 1997. 129, 133

T. Joachims. Making large scale SVM learning practical. *Advances in Kernel Methods - Support Vector Learning*, 1999. 78, 103

T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, 2002. 45

R. Johansson and P. Nugues. Extended constituent-to-dependency conversion for english. In *Proceedings of the 16th Nordic Conference on Computational Linguistics*, pages 105–112, 2007. 29

M. Johnson. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632, 1998. 49

S. Kahane. Bubble trees and syntactic representations. In T. Becker and H.-U. Krieger, editors, *Proceedings of Mathematics of Language (MOL5) Meeting*, pages 70–76, 1997. 21

S. Kahane. Grammaires de dépendance formelles et théorie Sens-Texte. In *Actes de la 8ème conférence sur le traitement automatique des langues naturelles*, volume 2, pages 17–76, 2001. 21, 24

R.M. Kaplan and J. Bresnan. Lexical-functional grammar: A formal system for grammatical representation. *Formal Issues in Lexical-Functional Grammar*, pages 29–130, 1982. 16, 20

T. Kasami. An efficient recognition and syntax-analysis algorithm for context-free languages. Technical report, Air Force Cambridge Research Lab, 1965. 28

R.S. Kayne. *French syntax: The transformational cycle*. MIT Press, 1975. 30

J. Kazama, S. De Saeger, K. Kuroda, M. Murata, and K. Torisawa. A Bayesian method for robust estimation of distributional similarities. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 247–256, 2010. 119

D. Klein and C.D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430, 2003. 49

T. Koo and M. Collins. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11, 2010. 53

T. Koo, X. Carreras, and M. Collins. Simple semi-supervised dependency parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 595–603, 2008. 120, 125, 140

S. Kübler. The PaGe 2008 shared task on parsing German. In *Proceedings of the Workshop on Parsing German*, pages 55–63, 2008. 52

S. Kübler, R. McDonald, and J. Nivre. Dependency Parsing. *Synthesis Lectures on Human Language Technologies*, 1(1):1–127, 2009. 2, 22, 50, 51, 57, 61

J. Le Roux, B. Favre, G. Mirroshandel, and A. Nasr. Modèles génératif et discriminant en analyse syntaxique : Expériences sur le corpus arboré de Paris 7. In *Actes de la 18ème conférence sur le traitement automatique des langues naturelles*, pages 371–382, 2011. 90

D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 768–774, 1998. 115, 118, 119, 121, 123, 131, 149

X. Lin, Y. Fan, M. Zhang, X. Wu, and H. Chi. Refining grammars for parsing with hierarchical semantic knowledge. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1298–1307, 2009. 120

D.M. Magerman. *Natural language parsing as statistical pattern recognition*. PhD thesis, Stanford University, 1994. 29

D.M. Magerman. Statistical decision-tree models for parsing. In *Proceedings of the 33rd annual meeting of the Association for Computational Linguistics*, pages 276–283. Association for Computational Linguistics, 1995. 26

C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999. 118

M.P. Marcus, M.A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. 29

D. McCarthy and J. Carroll. Disambiguating nouns, verbs, and adjectives using automatically acquired selectional preferences. *Computational Linguistics*, 29(4): 639–654, 2003. 144

D. McCarthy, R. Koeling, J. Weeds, and J. Carroll. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pages 279–286, 2004. 128, 129, 133

D. McClosky and E. Charniak. Self-training for biomedical parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 101–104, 2008. 100

D. McClosky, E. Charniak, and M. Johnson. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 152–159, 2006. ii, iii, 88, 98, 99, 100, 110, 111

R. McDonald and J. Nivre. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 122–131, 2007. 56

R. McDonald and F. Pereira. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–88, 2006. 53, 90

R. McDonald, K. Crammer, and F. Pereira. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98, 2005. 49, 53, 54, 146

R. McDonald, K. Lerman, and F. Pereira. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Natural Language Learning*, pages 216–220, 2006. 54, 55

I. Melčuk. *Dependency Syntax: Theory and Practice*. The SUNY Press, Albany, NY, 1988. 21, 22, 24, 28

P. Mertens. Restrictions de sélection et réalisations syntagmatiques dans Dicovalence. Conversion vers un format utilisable en TAL. In *Actes de la 17ème conférence sur le traitement automatique des langues naturelles*, 2010. 147, 153

G.A. Miller and W.G. Charles. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28, 1991. 118

G.A. Miller, C. Leacock, R. Tengi, and R.T. Bunker. A semantic concordance. In *Proceedings of the workshop on Human Language Technology*, pages 303–308, 1993. 144

S.A. Mirroshandel, A. Nasr, and J. Le Roux. Semi-supervised dependency parsing using lexical affinities. *Proceedings of ACL 2012*, 2012. 146, 147

A. Nasr. Analyse syntaxique probabiliste pour grammaires de dépendances extraites automatiquement. Habilitation à diriger des recherches, Université Paris 7, 2004. 22

J. Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies*, pages 149–160, 2003. 2, 28, 49, 53, 57, 59

J. Nivre. Incremental non-projective dependency parsing. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT)*, pages 396–403, 2007. 59

J. Nivre. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553, 2008. ii, iii, 2, 57, 58, 59, 60, 61, 64, 72, 73

J. Nivre and J. Nilsson. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 99–106, 2005. 107

J. Nivre, J. Hall, J. Nilsson, G. Eryigit, and S. Marinov. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 221–225, 2006. 2, 55, 57, 74, 80

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, 2007a. 49

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135, 2007b. ISSN 1351-3249. 53, 64, 74, 81, 84

D. Ó Séaghdha. Latent variable models of selectional preference. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 435–444. Association for Computational Linguistics, 2010. 145, 162

M. Olteanu and D. Moldovan. PP-attachment disambiguation using large context. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 273–280, 2005. 91

P. Pantel and D. Lin. An unsupervised approach to prepositional phrase attachment using contextually similar words. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 101–108, 2000. 91

P. Pantel, E. Crestan, A. Borkovsky, A. Popescu, and V. Vyas. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 938–947, 2009. 119

F. Pereira, N. Tishby, and L. Lee. Distributional clustering of English words. In *Proceedings of the 31st annual meeting of the Association for Computational Linguistics*, pages 183–190, 1993. 118, 119

S. Petrov and R. McDonald. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language*, 2012. 50

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440, 2006. 120

S. Petrov, P.C. Chang, M. Ringgaard, and H. Alshawi. Uptraining for accurate deterministic question parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 705–713, 2010. 99

C.J. Pollard and I.A. Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, 1994. 16, 21

P. Resnik. A class-based approach to lexical discovery. In *Proceedings of the 30th annual meeting on Association for Computational Linguistics*, pages 327–329, 1992. 144

P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11(95):130, 1999. 91

A. Ritter, Mausam, and O. Etzioni. A latent dirichlet allocation method for selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 424–434. Association for Computational Linguistics, 2010. 145, 162

J.J. Robinson. Dependency structures and transformational rules. *Language*, pages 259–285, 1970. 26

F. Rosenblatt. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. 41

K. Sagae. Self-training without reranking for parser domain adaptation and its impact on semantic role labeling. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 37–44, 2010. 99

K. Sagae and A. Gordon. Clustering words by syntactic similarity improves dependency parsing of predicate-argument structures. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 192–201, 2009. 120, 125

B. Sagot. The Lefff, a freely available, accurate and large-coverage lexicon for French. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*, 2010. 53, 76

B. Sagot and D. Fišer. Building a free French wordnet from multilingual resources. In *Proceedings of Workshop on OntoLex*, pages 14–19, 2008. 127, 128

P. Sgall, E. Hajicová, and J. Panevová. *The meaning of the sentence in its semantic and pragmatic aspects*. Reidel, Dordrecht, 1986. 21

M. Shimbo and K. Hara. A discriminative learning model for coordinate conjunctions. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 610–619, 2007. 91

A. Sigogne and M. Constant. Using subcategorization frames to improve French probabilistic parsing. In *Proceedings of the 11th Conference on Natural Language Processing (KONVENS'12)*, 2012. 120

A. Sigogne, M. Constant, and E. Laporte. French parsing enhanced with a word clustering method based on a syntactic lexicon. In *Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages*, pages 22–27, 2011. 120

F. Smadja. Retrieving collocations from text: Xtract. *Computational linguistics*, 19 (1):143–177, 1993. 119

M. Steedman, M. Osborne, A. Sarkar, S. Clark, R. Hwa, J. Hockenmaier, P. Ruhlen, S. Baker, and J. Crim. Bootstrapping statistical parsers from small datasets. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 331–338, 2003. 98

L. Tesniere. *Eléments de syntaxe structurale*. Klincksieck, Paris, 1959. 21, 22

J. Tiedemann. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In *Recent Advances in Natural Language Processing*, volume 5, pages 237–248. John Benjamins, Amsterdam, 2009. 104

P.D. Turney and P. Pantel. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188, 2010. ISSN 1076-9757. 117

L. Van Der Plas and G. Bouma. Syntactic contexts for finding semantically related words. In *Computational linguistics in the Netherlands*, 2004. 119

G. Van Noord. Using self-trained bilexical preferences to improve disambiguation accuracy. In *Proceedings of the 10th International Conference on Parsing Technologies*, pages 1–10, 2007. 145

P. Vossen, editor. *EuroWordNet: a multilingual database with lexical semantic networks*. Kluwer Academic Publishers, Dordrecht, 1998. 127

D. Xiong, S. Li, Q. Liu, S. Lin, and Y. Qian. Parsing the Penn Chinese Treebank with semantic knowledge. In *Proceedings of the International Joint Conference on Natural Language Processing*, pages 70–81, 2005. 119

H. Yamada and Y. Matsumoto. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies*, pages 195–206, 2003. 2, 26, 28, 29, 49, 53, 59, 107

D.H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and control*, 10(2):189–208, 1967. 28

Y. Zhang and S. Clark. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 562–571, 2008. 163

Y. Zhang and J. Nivre. Transition-based parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 188–193, 2011. 163

G. Zhou, J. Zhao, K. Liu, and L. Cai. Exploiting web-derived selectional preference to improve statistical dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 1556–1565, 2011. 146